Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE FI DE CARRERA

TÍTOL: Carrier Management, versión para dispositivos móviles

AUTOR: David Vinagre Borràs

TITULACIÓ: Enginyeria técnica en Informàtica de gestió. Escola Politècnica Superior d'Eninyeria de Vilanova i la Geltrú (EPSEVG)

DIRECTOR: Neus Català Roig

DEPARTAMENT: Departament de Ciències de la Computació (CS)

DATA: 01-09-2015

TÍTOL: Carrier Management, versión para dispositivos móviles

COGNOMS: Vinagre Borràs **NOM: David**

TITULACIÓ: Enginyeria técnica en Informàtica de gestió

ESPECIALITAT: Informática de gestió **PLA: 92**

DIRECTOR: Neus Catalá Roig

DEPARTAMENT: CS

QUALIFICACIÓ DEL PFC

TRIBUNAL

PRESIDENT **SECRETARI** **VOCAL**

DATA DE LECTURA:

Aquest Projecte té en compte aspectes mediambientals: □ Sí □ No

PROJECTE FI DE CARRERA

RESUM (màxim 50 línies)

La finalidad de este PFC es crear una aplicación nativa para smartphones con sistema operativo Windows Phone que proporcione a los usuarios (Transportistas) la posibilidad de gestionar los viajes que le han sido propuestos o asignados. Se pretende que el uso de la aplicación sea sencillo y claro. Priorizando una navegación intuitiva.

La idea del proyecto surge del ámbito laboral, a raíz de algunas implantaciones de sistemas de gestión de transporte consideré que ciertas necesidades de los clientes no estaban bien cubiertas. No obstante, el proyecto no tiene pretensiones de ser una aplicación comercializable.

El proyecto ha arrancado con la etapa de análisis. Inicialmente me he marcado los siguientes objetivos:

- Aprender el lenguaje C#
- Aprender a programar en smartphones Windows.
- Aprender a crear un cliente que se comunique con un SOAP Web Service.

Tras tener los objetivos definidos, he analizado cuales serían los requisitos que debería garantizar la aplicación, así como elementos que serían necesarios. Del análisis de estos elementos, he seleccionado más adecuados.
Con esto aclarado, las siguientes semanas han transcurrido con el diseño las pantallas que conforman la aplicación, revisión de los datos serán necesarios extraer y cual sería el modelo conceptual. Esta parte la he ido revisado a medida que iba descubriendo nuevos controles de Layout en la fase de implantación (Por ejemplo el control Hub).

El resultado ha resultado satisfactorio, una aplicación que permite consultar tanto viajes aceptados, como propuestos, permitiendo a los transportistas aceptarlos, ver los detalles de cada viaje, situando en un mapa las paradas y permitiendo confirmar las mismas.

La realización de este proyecto me ha permitido refrescar los conocimientos adquiridos en la carrera que creo poder sacar partido laboralmente.

Paraules clau (màxim 10):

Windows Phone	Web Services	TMS	SOAP
C#	XAML	MapControl	Visual Studio 2013

Agradecimientos

Quiero mostrar mi agradecimiento a mis padres por su continuo esfuerzo y apoyo en todas decisiones que he tomado en mi vida, a Marylin por su amor y comprensión, especialmente mientras he estado invirtiendo nuestro tiempo libre en este proyecto, a Neus por confiar en mí y estar siempre pendiente de la finalización del proyecto y a todos los compañeros de trabajo que me han aportado nuevos conocimientos y han resuelto todas mis dudas.

1. INTRODUCCIÓN

1.1. Justificación del proyecto

Para entender los motivos que me han llevado a plantear este proyecto es necesario recapitular la historia reciente de JDA Software, una empresa estadounidense especializada en todo tipo de soluciones informáticas para la gestión, mejora y optimización de los diferentes puntos de la cadena de suministro.

Entre cada uno de los puntos de la cadena de suministro se producen unos flujos de transporte y es por ello de la necesidad de disponer de un sistema de gestión del transporte, también conocido como TMS (Transportation Management System). Son un tipo de aplicaciones que tienen por objetivo cubrir el ciclo de transporte que se genera en cualquier empresa.

En 2006 JDA Software adquirió otro de los principales proveedores de software relacionado con la cadena de suministro, Manugistics, adquiriendo así uno de sus productos estrella, Manugistics Transport.
Durante años ha sido uno de los TMS de referencia y como consecuencia de la alianza entre JDA y la empresa para la cual trabajo, Agora Europe, fue el TMS que se estuvo implantando en múltiples clientes.

Pasados los años, JDA Software repitió la operación con la compra de i2 Technologies, en el año 2010. La adquisición reunió a las dos empresas líderes del sector, cada una con su propio TMS.

Como consecuencia JDA se vio en obligada a focalizar el desarrollo de una de las dos herramientas y decidió apostar por el TMS de i2, Transportation Manager y no continuar con el desarrollo de Manugistics Transport. Por otra parte, el hecho de que este último continuara con una arquitectura tipo cliente – servidor y que ofreciera muy poca flexibilidad para comunicarse con otros sistemas fueron motivos determinantes en la decisión de apostar por el TMS de i2, un sistema más complejo y versátil.

Llegados a este punto es donde surge el primer motivo para crear una aplicación que se comunique con Transportation Manager.
En las primeras implantaciones del sistema en clientes no hemos podido sacado provecho de disponer de un servicio web SOAP al no tener adquiridos aun los conocimientos. Debido a ello nos hemos encontrado que en algunos proyectos los propios clientes se han mostrado interesados en crear las interfaces de entrada / salida vía servicios web y se ha tenido que descartar esta opción.

La siguiente razón que me motivó a plantear este proyecto es el hecho que actualmente Transportation Manager no ofrece una aplicación móvil para los transportistas, estos realizan su operativa desde una aplicación web, incluso tienen un módulo especifico llamado Carrier Equipment Availavility donde actualizar los recursos que disponen para realizar los viajes

que se les han asignado pero no ha sido diseñado para visualizarse desde un dispositivo móvil.

Debido a todo esto, me parecía interesante crear una aplicación para dispositivos móviles y así dotar a los transportistas del acceso inmediato a información de, por ejemplo, viajes aceptados, viajes propuestos, confirmaciones de carga y gestión de albaranes.

Además, el abaratamiento del precio medio de terminales inteligentes así como la incorporación de cámara, gps, y APIs específicas de cartografía pueden hacer que la experiencia de usuario sea muy superior a la ofrecida por la aplicación web.

1.2. Objetivos del Proyecto

El principal objetivo de este proyecto final de carrera no es otro que el de adquirir nuevos conocimientos y poner en práctica la metodología aprendida en el trascurso de los cursos universitarios.

Siendo un poco más concreto los objetivos serian:

- Aprender el lenguaje C#
- Aprender a programar en smartphones Windows.
- Aprender a crear un cliente que se comunique con un servicio web SOAP Web.
- Diseñar e implementar una aplicación para dispositivos móviles.

Para conseguir cumplir con los objetivos descritos, pretendo crear un servidor virtual en el cual tener instalado el TMS y el servicio web. En otro equipo preparar un ambiente de trabajo con las herramientas desarrollo y emulador para poder testear la aplicación.

Dicha aplicación móvil espero que sirva para que todo transportista dado de alta en el TMS pueda ver los detalles de los viajes que los gestores de transporte le han ofertado y los viajes que ya ha aceptado. Puesto que puede acceder a los detalles, el transportista podrá aceptar o rechazar dichos viajes propuestos.
Además de los viajes aceptados, podrá acceder a la información de las paradas de carga o descarga e ir confirmando las paradas, una vez confirmadas las paradas el viaje podrá ser cerrado.
Por último tengo intención de marcar en un mapa las paradas del viaje para que sirva de ayuda al transportista.

Dado que la aplicación debería ser utilizada por un colectivo de personas que no necesariamente debe saber manejarse con nuevas tecnologías, a la hora de su diseño debo intentar una interfaz clara y sencilla.

2. EVALUACIÓN TECNOLOGICA

2.1. Sistemas operativos para dispositivos móviles

2.1.1. Evolución de los Sistemas Operativos

Un sistema operativo móvil [002] es un sistema operativo (SO) diseñado para controlar dispositivos tales como smartphones, tablets, PDAs o cualquier otro dispositivo móvil.
Hay que tener en cuenta que durante los últimos años estos dispositivos cada vez son más complejos y es por ello que los SO móviles han ido añadiendo cada vez más características como son el control de pantallas táctiles, video cámaras, reconocimiento de voz, conectividad inalámbrica por Bluetooth, Wi-Fi, UMTS, LTE y NFC, por poner algunos ejemplos.

Además, fruto de la constante evolución tecnológica estos dispositivos ya son capaces de incluir características propias de los SO de ordenadores personales [001].

Durante las diferentes etapas de estos dispositivos han ido apareciendo múltiples SO, es por ello que nos centraremos en los SO más relevantes en la actualidad:

- Android [004]: Es un SO desarrollado por Google, anunciado en el año 2007 y basado en el kernel de Linux, gran parte de su código es gratuito y de código libre, código escrito en C/C++ y Java.
  El desarrollo de aplicaciones para Android se hace habitualmente en Java y con el Android SDK.
  Actualmente la plataforma Android es una de las preferidas para la comunidad desarrolladora.

- iOS [003]: Es el SO de la empresa Apple para dispositivos móviles. Fue anunciado durante la MacWorld Conference del año 2007, iOS deriva de Mac OS X, que a su vez está basado en Darwin BSD (Unix), es un SO propietario y de código cerrado, dicho código está escrito en C/C++, Objective–C y Swift.
  Apple proporciona un SDK que entre otras herramientas incluye un simulador y su propio IDE llamado Xcode. Cabe señalar que para iniciarse en el desarrollo de apps para iOS es necesario registrarse previamente en el iOS Development Program.

- Windows Phone [005]: Es la enésima apuesta de Microsoft dentro del segmento de los dispositivos móviles tras Windows Mobile. Se anunció en Octubre de 2010 buscando captar el segmento del gran público, su kernel está basado en Windows NT. Es un SO propietario y de código cerrado, dicho código está escrito en C/C++.
  El desarrollo de aplicaciones para Windows Phone se basa en el SDK proporcionado y en el uso de Visual Studio, el IDE oficial. El lenguaje de programación para la interfaz de usuario es XAML y para el código puede ser C# o Visual Basic.

- BlackBerry OS: Es el SO propietario desarrollado por Blackberry para los dispositivos móviles de la compañía.
  Nació en 1999 aunque la última versión del SO, bautizada como Blackberry 10, data de 2013. Es un SO cuyo nicho de mercado tradicionalmente ha sido el empresarial.
  Blackberry OS deriva del SO QNX (Sistema en tiempo real del tipo UNIX).
  El código está escrito en C/C++, Cascades, HTML5 y Adobe AIR. Blackberry ofrece un SDK con las herramientas necesarias para portar apps de otros sistemas o para la creación desde cero mediante el IDE Momentics.

- Ubuntu Touch: Es un SO basado en Linux desarrollado por Canonical.
  Presentado en 2013, ofrece una interfaz llamada Unity, común en todos los dispositivos móviles y equipos sobremesa.
  Escrito en HTML5, C/C++ y QML, el desarrollo de nuevas apps es posible mediante el SDK proporcionado por Canonical.

- Firefox OS: Es un SO desarrollado por Mozilla Corporation en colaboración con otras empresas, de código abierto y basado en el núcleo de Linux.
  Su código está escrito en HTML5, JavaScript y C++. El desarrollo de aplicaciones es gestionado por su comunidad, son aplicaciones creadas con HTML5, CSS y JavaScript.

- Sailfish OS: Es un SO de código propietario aunque con componentes de código abierto, desarrollado por la empresa Jolla Ltd, una empresa creada por ex-empleados de Nokia.
  Sailfish OS nació en el 2013 y está basado en el kernel de Linux.

2.1.2. Análisis de mercado

El mercado de los dispositivos móviles actualmente es uno en los que las empresas de electrónica de consumo más están invirtiendo y de los más dinámicos desde hace prácticamente una década, no obstante, figuras relevantes del sector han denominado este periodo era Post-PC ya que los hábitos de consumo del usuario medio están variando de manera significativa y un único dispositivo, que se puede llevar a cualquier lugar, tiene la capacidad de procesamiento y velocidad de acceso a datos suficiente para las tareas más habituales del usuario doméstico.

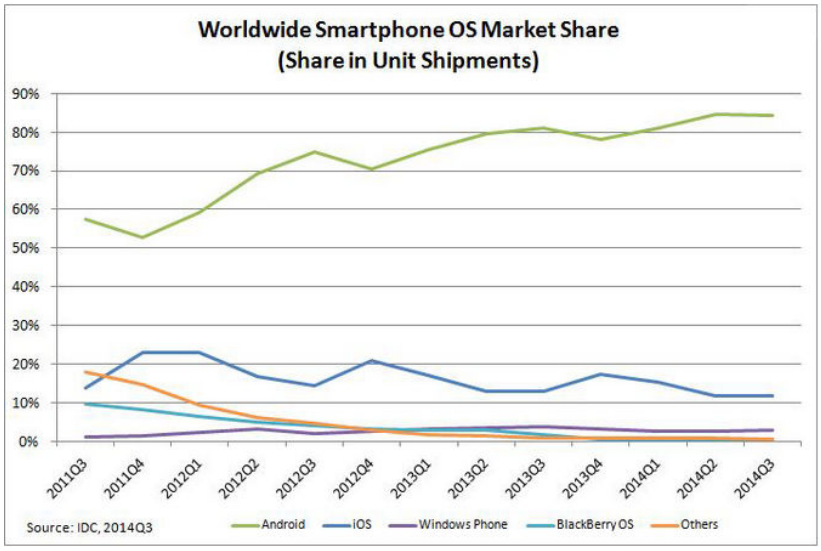
Según los datos del siguiente gráfico [006], las ventas de smartphones han aumentado un 20% respecto al tercer trimestre de 2013 y es el primer año en el que el número de smartphones supera el de teléfonos móviles.

Company	3Q14 Units	3Q14 Market Share (%)	3Q13 Units	3Q13 Market Share (%)
Samsung	73,212.4	24.4	80,356.8	32.1
Apple	38,186.6	12.7	30,330.0	12.1
Huawei	15,934.9	5.3	11,665.7	4.7
Xiaomi	15,772.5	5.2	3,617.5	1.5
Lenovo	15,011.9	5.0	12,882.0	5.2
Others	142,891.6	47.5	111,445.0	44.5
Total	**301,009.9**	**100.0**	**250,297.0**	**100.0**

1: Ventas mundiales de Smarthpones en el 3er trimestre 2013-2014(datos en miles). Fuente: www.gartner.com

Uno ciclo de vida más corto que en otros aparatos electrónicos y la constante evolución de las tecnologías asociadas hacen que hayamos visto numerosos cambios en el sector y, aunque en menor medida, también en los SO de referencia.

Si nos centramos en la actualidad, el SO de referencia es Android, desde su aparición ha tenido un crecimiento casi exponencial y la mayor parte de los fabricantes lo han adoptado como SO en sus dispositivos de gama alta.



Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q3 2014	84.4%	11.7%	2.9%	0.5%	0.6%
Q3 2013	81.2%	12.8%	3.6%	1.7%	0.6%
Q3 2012	74.9%	14.4%	2.0%	4.1%	4.5%
Q3 2011	57.4%	13.8%	1.2%	9.6%	18.0%

2: Cuota de mercado de Smartphones según SO. Intervalo 2011-2014 Fuente: www.idc.com

La figura anterior, según los datos de la consultora IDC [007], así lo constata.
Android mantiene un amplio margen respecto a iOS, cuya cuota de mercado va fluctuando ligeramente ya que Apple suele ganar cuota de mercado los meses inmediatamente posteriores a la venta de sus nuevos dispositivos.

En lo relativo a Windows Phone, parecen estar viviendo una situación un tanto agridulce, se consolidan como tercera plataforma más utilizada pero el margen con los dos grandes SO se acentúa. Se espera que la nueva versión de su SO (Windows 10), así como una nueva estrategia de marketing le hagan ganar cuota de mercado.

2.2. Entornos de programación

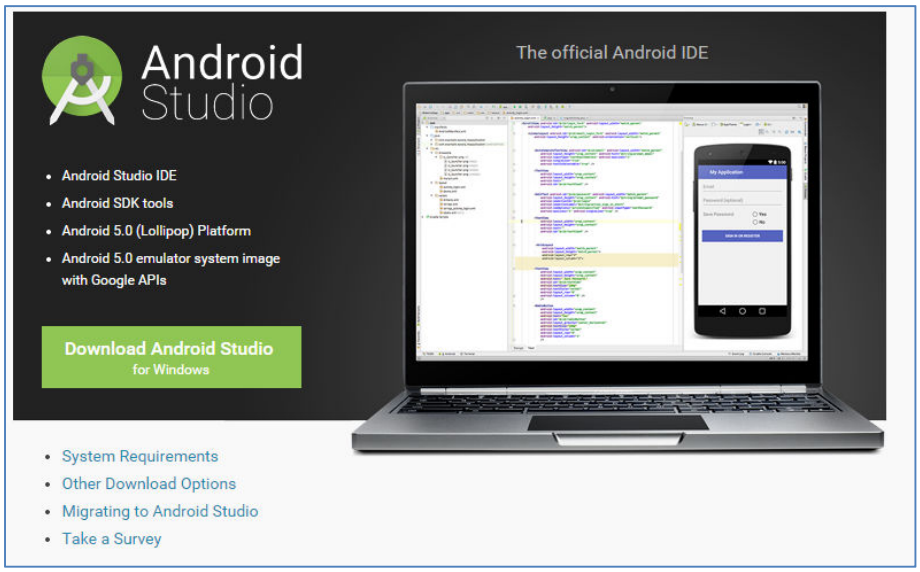
En este apartado se pretende hacer una pequeña descripción de las herramientas de programación que los proveedores proporcionan para el desarrollo de aplicaciones, centrándonos en los 3 SO principales.

2.2.1. Android

Las aplicaciones para Android normalmente están desarrolladas en Java y gracias a la máquina virtual Dalvik , una máquina virtual nacida a partir de la Java Virtual Machine (JVM) las aplicaciones se pueden ser ejecutadas y portadas a muy diverso hardware.

Para ello se utiliza el Android Software Development Kit (SDK) este es el SDK oficial y el que contemplare en esta fase de análisis, aunque es posible la creación de aplicaciones con otros entornos de desarrollo.
El SDK de Android incluye una serie de herramientas de desarrollo, a grandes rasgos son: un depurador de código, bibliotecas base, un emulador de teléfono basado en QEMU, documentación, códigos de ejemplo y tutoriales.
Es posible su instalación en equipos bajo Linux, Mac OS X y Windows XP o superior. Actualmente también se pueden desarrollar aplicaciones en equipos con Android utilizando la aplicación *AIDE – Android IDE – Java, C++* y el editor *Java editor.*

El IDE (Integrated Development Environment) utilizado por defecto es el Android Studio [008] (Como se muestra en la siguiente figura), este ha pasado a substituir a Eclipse como IDE preferente. Android Studio aporta como beneficios una instalación sumamente sencilla y además evita tener que instalar plugins externos.



3: Detalle Android Studio

Otros IDE disponibles serían, el ya mencionado Eclipse (utilizando el plugin Android Development Tools) NetBeans o IntelliJ IDEA, incluso los desarrolladores podrían

utilizar cualquier editor de texto para crear/editar los ficheros Java y XML y posteriormente crear, compilar o ejecutar las apps vía terminal.

2.2.2. iOS

Con la idea de permitir a terceros desarrollar aplicaciones para iOS, Apple puso a la disposición de la comunidad el iOS SDK, en el encontraremos las herramientas, documentación y bibliotecas básicas para poder inicializarnos en el desarrollo de aplicaciones.

Los componentes del SDK son los siguientes:

- Cocoa Touch framework: Es un framework escrito en Objective-C que tiene por objetivo simplificar el uso de hardware y recursos propios de los dispositivos móviles de Apple y que no se encuentran en Mac OS X.

- Xcode: Xcode [009] es el IDE creado por Apple que trabaja de manera conjunta con Interface Builder.
  Este conjunto de herramientas además incluye una colección de compiladores del proyecto GNU (GCC) y puede compilar código C, C++, Objective-C, Objective-C++, AppleScript y Java.

4: Detalle contenido Xcode.

- Simulador iOS: Es una aplicación integrada en Xcode, nos permite ir verificando del funcionamiento y visualizando de una manera más automática los cambios y mejoras de nuestro desarrollo.

- Instrumentos de análisis
- Interface Builder: Es una aplicación integrada en Xcode, su principal objetivo es el de ayudar a los desarrolladores en el diseño de interfaces.

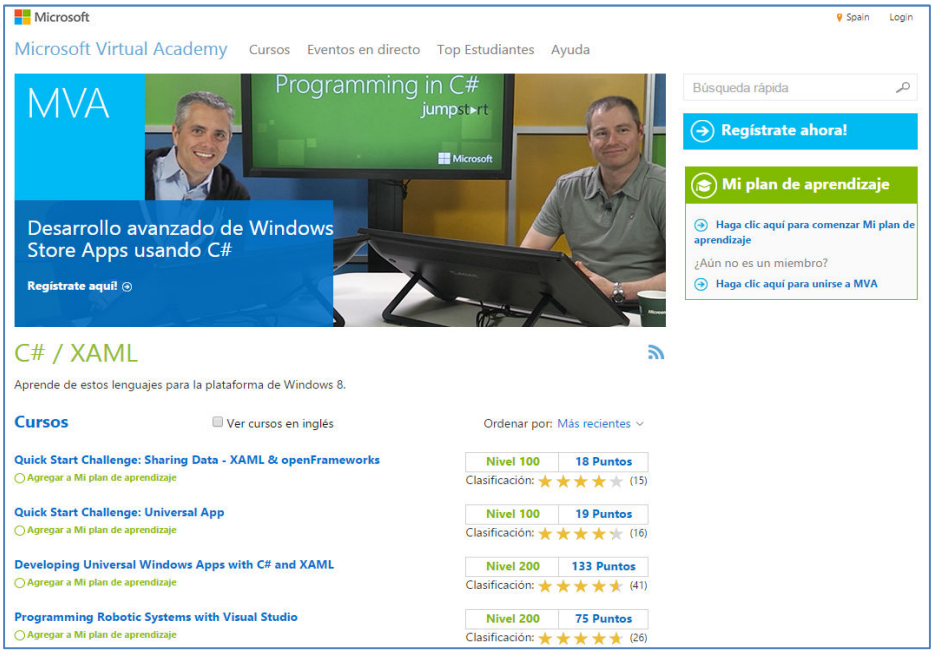
El lenguaje principal para el desarrollo de aplicaciones es Objective-C, una superclase de C que ha sido el lenguaje principal desde que Apple adquirió NeXT en 1996 y tras la creación del SO Mac OS X. No obstante, en la Worldwide Developers Conference de 2014 se presentó un nuevo lenguaje para iOS llamado Swift. Es un lenguaje compilado y multiparadigma, según la propia empresa mejora en diferentes aspectos a Objective-C ya que tiene una mejor curva de aprendizaje y mayor velocidad de ejecución de las aplicaciones.

Uno de los puntos que más controversia crea entre los desarrolladores es el hecho de que las herramientas de desarrollo solo están disponibles para Mac OS X.

2.2.3. Windows Phone

De la misma manera que sus principales competidores, Microsoft facilita a la comunidad desarrolladora una serie de herramientas con las que crear apps para la Windows Store.

Además, conscientes de la importancia de tener una tienda de aplicaciones lo más completa posible, Microsoft ofrece un portal de video tutoriales y bibliografía bastante extensa, es la Microsoft Virtual Academy (http://www.microsoftvirtualacademy.com). Podemos ver un detalle de los cursos en la siguiente figura.



5: Ejemplo recursos ofrecidos en la MVA

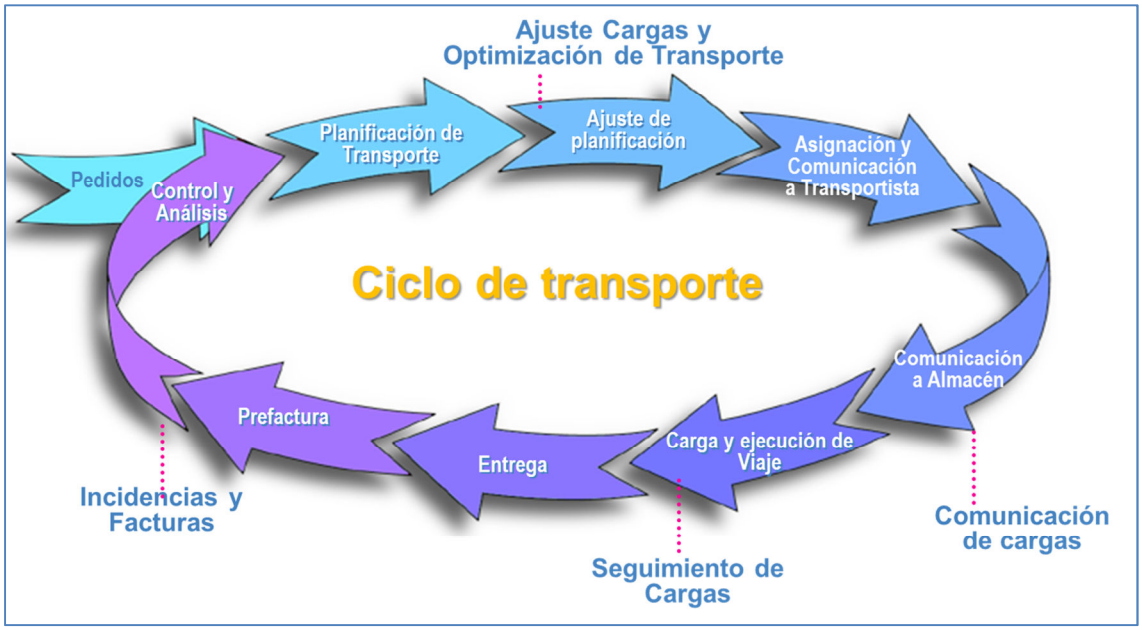
Centrándome de nuevo en las herramientas, el pilar principal del que disponemos es el Visual Studio. Es el IDE oficial desarrollado y distribuido por Microsoft. Soporta lenguajes de programación tales como Visual C++, C#, Visual Basic.NET, Java, Python, Ruby y PHP.
A partir de la versión 2005 Microsoft ofrece gratuitamente unas versiones denominadas '*Express'* y orientadas a principiantes, estudiantes y aficionados a la programación.

Además del IDE, necesitaremos disponer del Windows Phone SDK, este kit de desarrollo contiene varios emuladores, bibliotecas, plantillas y herramientas de diseño (Microsoft Blend).

2.3. Sistemas de gestión del transporte

Un sistema de gestión del transporte [010], también conocido como TMS (Transportation Management System) es un software que se pretende encargar de la logística alrededor del transporte, normalmente se sitúa entre el ERP y el módulo encargado de la distribución y/o gestión de almacén.

En un escenario típico las órdenes de transporte llegarían al sistema y serian evaluadas por el planificador en términos de coste, tiempo, capacidad de carga, etc.
Posteriormente el planificador ofrecerá al usuario varias soluciones de ruta que serán validadas por él y una vez seleccionada la mejor opción, el sistema ofertará la carga al trasportista. Si este último está interesado en transportar la carga, puede entrar al sistema y aceptar el viaje propuesto.
A partir de aquí el transportista puede ir informando las recogidas, entregas e incidencias en el viaje.



6: Ciclo de transporte

Por último, una vez el viaje se ha completado, mediante el módulo financiero se puede generar la factura proforma al transportista. Todas estas fases están descritas en la figura anterior.

Estos sistemas pueden ser implementados en diversas modalidades:

1. Ad-Hoc: El modelo consistiría en un exhaustivo análisis de los procesos y metodologías de la cadena de suministro de la empresa, para finalmente diseñar un software a medida que se adaptará exactamente a esas necesidades.

2. Hosted Licensing: Este modelo consiste en implementar uno de los diferentes TMS creados por empresas de desarrollo de software en un servidor de la propia compañía y parametrizarlo según los procesos del departamento de transporte.

3. SaaS: Software as a Service [012], es un modelo en el cual el TMS está desplegado sobre la infraestructura de un proveedor de servicios. Reduciendo los costes en recursos y gestión del departamento de sistemas de la compañía.

Así pues, dejando de lado el software a medida, en el mercado ofrece diversas soluciones para la gestión del transporte, haré un breve repaso de los 3 principales.

2.3.1. Oracle

Oracle Corporation, además de ser mundialmente conocida por su sistema de gestión de base de datos relacional, ofrece un portfolio de aplicaciones dedicadas a la gestión de la cadena de suministro bastante completo.

El TMS desarrollado por Oracle es conocido como Oracle Transportation Management [013] y sus funciones principales son:

- Transportation Order Management: Integración con los sistemas externos de gestión de pedidos con el objetivo de gestionar la demanda de transporte interna.

- Rate Management: Un repositorio global de tarifas y motor de tarificación a partir de los datos relativos a los servicios y tarifas ofrecidas por los transportistas.
- Shipment Management: Planificación y ejecución de envíos maximizando la eficiencia. Automatización de la toma de decisiones asistida para un amplio rango de escenarios. Manejo de excepciones durante la ejecución del transporte.

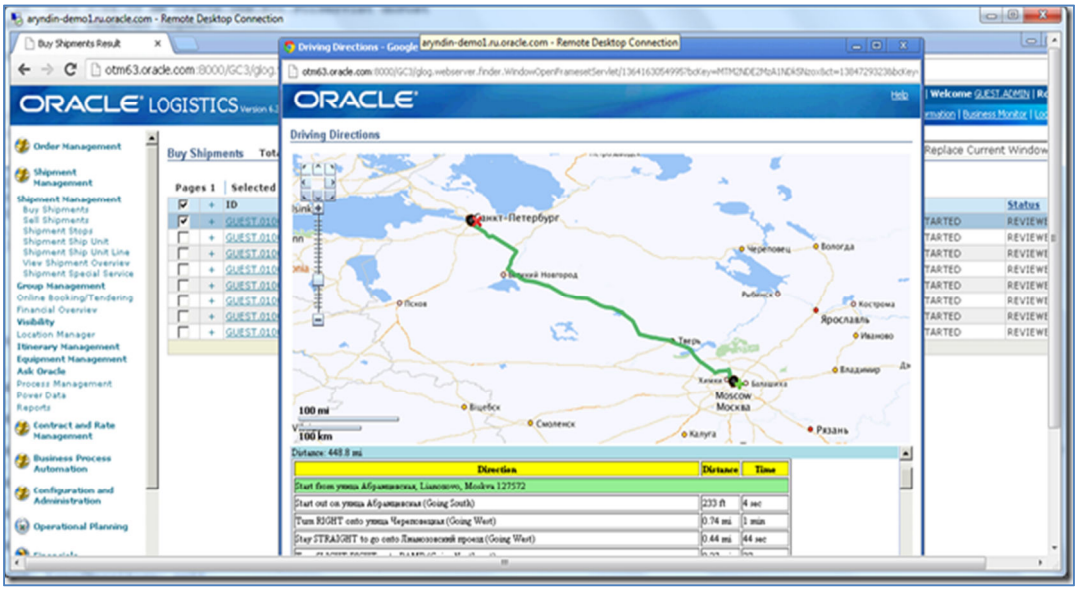
- Booking and Tendering: Permite la colaboración con los proveedores de los servicios de transporte de múltiples maneras (email, web, dispositivos móviles y XML). Permite compartir los planes de ejecución y gestionar las respuestas de los propios transportistas, incluyendo un proceso de oferta público.

- Event Management: Gestionar de manera proactiva el ciclo de vida de los pedidos y los envíos mediante un monitor de estados. Procesar los pedidos cuando las confirmaciones de recogida no han sido recibidas en un intervalo de contingencia establecido. Recepción de actualizaciones por parte de los transportistas.

- Business Process Automation: Maximizar la productividad mediante la automatización de procesos.

- Reporting & Document Management: Creación de informes y documentos mediante el Oracle Business Intelligence Publisher

El TMS de Oracle se basa en una arquitectura web (ver figura), proporcionando flexibilidad y escalabilidad tanto a los usuarios como a los transportistas ya que ofrece además soporte para dispositivos móviles.



6: Captura de la interfaz de OTM

2.3.2. SAP

Fundada en 1972 por 5 exingenieros de IBM, la empresa con sede en Walldorf (Alemania) desde sus inicios ha centrado su actividad en el desarrollo de aplicaciones de gestión empresarial, en concreto es mundialmente conocida por sus aplicaciones en las áreas de CRM (Customer Relationship Management), ERP (Enterprise Resource Planning) y SCM (Supply Chain Management).

Dentro del área SCM, SAP dispone de su propia solución para la gestión del transporte, es el SAP Transportation management [015] y las áreas del proceso que cubre son las siguientes:

- Transportation Requirements Management: El software de SAP soporta un proceso de gestión que integra el procesamiento de pedidos de transporte con las órdenes de

pago y cobro. Se pueden gestionar requerimientos de transporte recibidos de forma manual o electrónicamente, independientemente del origen de datos.

- Freight Planning and Optimization: Con el software de SAP es posible planificar, consolidar y optimizar envíos nacionales e internacionales multimodo, considerando limitaciones y restricciones, costes y multas. Los planes de optimización ayudan a reducir el gasto en transporte, manteniendo los niveles esperados de servicio al cliente.

- Freight Tendering: La aplicación posibilita ofertar un envío a uno o varios transportistas utilizando reglas de negocio automatizadas. Las asignaciones de transportistas pueden están basadas en un sistema optimizado de rankings o hacer uso de una asignación por preferencia según la información del transportista. Estas automatizaciones ayudan a reducir los costes operativos y mejorar la productividad.

- Freight Order Forwarding: El software permite gestionar y liquidar contratos de transporte ya sea hacia los transportistas o hacia cliente final. Las capacidades para gestionar los costes soportan acuerdos de transporte de mercancías, así como tarifas o cargos extra. Puede resolver disputas de manera rápida y de forma colaborativa. Estas capacidades se pueden integrar con el software de finanzas para tareas de auditoria, pago al portador, facturación de clientes o distribución de costes.

- Transportation Execution and Monitoring: Las operaciones de transporte son fundamentales a la hora de poner en marcha los planes de transporte. La clave de estas actividades es la gestión de la comunicación con los transportistas, la notificación y seguimientos de los pedidos, y la gestión de los documentos asociados a esta, incluyendo impresiones e informes.

2.3.3. JDA Software

JDA Software es una empresa estadounidense dedicada a la consultoría y desarrollo de software, fundada en 1985 y actualmente con sede en Scottsdate, Arizona.

Su negocio principal está relacionado con la creación e implantación de aplicaciones especializadas en diversos ámbitos de la gestión de la cadena de suministro como pueden ser la planificación de la fabricación, gestión de la demanda, aprovisionamiento, gestión de almacenes, planogramación de los puntos de venta o gestión del transporte.
El portfolio total de JDA es de unas 30 aplicaciones.

7: Portfolio de soluciones ofrecidas por JDA

Una de las aplicaciones referencia de JDA es el JDA Transportation Manager [016] y se caracteriza por ofrecer soporte integral en el ciclo de vida de transporte, desde la gestión de pedidos mediante el servicio de atención al cliente hasta la facturación financiera. Facilitando la colaboración con clientes y transportistas.
Además permite la gestión del proceso de transporte y proporciona visibilidad extremo a extremo en todas las etapas.

JDA Transportation Manager es un componente clave dentro de la suite de transporte, integrándose con JDA Transportation Planner, un potente planificador.
El TMS consolida pedidos, evalúa alternativas y crea planes óptimos, considerando los compromisos de entrega al cliente, la disponibilidad de equipamientos y otras restricciones del mundo real.

La solución permite supervisar de manera proactiva su operativa basándose en un modelo de excepciones, proporcionando visibilidad a través de expedidores, transportistas, vendedores y clientes.

Es una sofisticada herramienta de gestión que puede manejar altos volúmenes de pedidos, flujos de información basados en la web y garantizar que cualquier replanificación debida a circunstancias imprevistas se puede lograr fácilmente con mínima intervención del usuario.

2.4. Tecnologías elegidas

En este apartado se pretende justificar los motivos que me han llevado a elegir las tecnologías necesarias para el desarrollo de la aplicación.

2.4.1. SO: Windows Phone

Tras analizar brevemente los diferentes SO para smartphones y centrarme en profundidad en los 3 de mayor relevancia (Windows Phone, Android, iOS) he decidido elegir desarrollar mi aplicación para que se ejecute sobre Windows Phone.

Como he descrito anteriormente, es una plataforma con una cuota de mercado no tan elevada como sus competidores en valores absolutos, aunque en terminales de gama de entrada la cuota de mercado mejora, el rendimiento de dichos terminales parece mejorar en mucho al de competidores como Android.

Además creo que es una plataforma que va a crecer en los próximos años y en la que Microsoft está haciendo grandes esfuerzos para apoyarla. De hecho ya hay una nueva versión del SO prevista para este año y con la cual Microsoft pretende unificar sus plataformas.

Pretendo asimismo aprovechar las sinergias que me puede proporcionar el aprendizaje de C# y XAML para nuevas oportunidades en el ámbito profesional.

2.4.2. IDE: Visual Studio

Los motivos para elegir Visual Studio son algo obvias dado que es el IDE oficial de Microsoft para desarrollar aplicaciones bajo XAML, C#. Otro detalle es que integra de manera adecuada el emulador con el editor y compilador.

Además con la versión Express, podemos desarrollar aplicaciones de manera gratuita con fines educativos.

2.4.3. TMS: JDA Transportation Manager

Dentro de los sistemas de gestión de transporte analizados, he decidido trabajar con la solución creada por JDA.

La decisión está basada en gran medida por mi experiencia laboral, ya que he estado involucrado en los últimos años en tareas de instalación y configuración de dicho TMS con lo que el tiempo necesario para preparar el entorno de trabajo será menor.

2.4.4. Web Service: SOAP

El diseño de la aplicación requerirá de un servicio web que se encargue intercambiar la información entre el TMS y la aplicación desarrollada. No he llegado a hacer un análisis en profundidad de las diferentes opciones en el mercado ya que tras decidir el TMS, implícitamente se está eligiendo el tipo de servicio web.

Concretamente con JDA Transportation Manager se despliega un servicio web basado en el protocolo SOAP [020] (Simple Object Application Protocol).

3. DISEÑO DE LA APLICACIÓN

3.1. Requisitos previos

Existen una serie de tareas a cumplir antes de iniciar el desarrollo de la aplicación, es necesario adecuar el entorno de trabajo desde donde se realizará el proyecto con los programas y herramientas necesarios para dicho fin.
Además, ya que la aplicación a desarrollar va a requerir comunicarse e intercambiar mensajes con el TMS, será necesario crear un servidor virtual con una instancia del mismo y una base de datos Oracle.

Así pues, los requisitos previos son:

- Oracle Weblogic 12.1 [020]: Es un servidor de aplicaciones web y java, es necesario para desplegar el desarrollo web del TMS.
- Oracle Database 11g: Es el motor de base de datos relacional bajo el que se almacenarán los datos del TMS.
- JDA Transportation Manager 8.1: El TMS elegido con él cual la aplicación a desarrollar intercambiará mensajes.
- JDA CIS 8.1: Componente de JDA, tiene por objetivo crear una capa común de integración entre las soluciones de la empresa. Sobre esta capa se creará el servicio web.
- Visual Studio Express 2013: El IDE donde se instalan las herramientas de desarrollo de Windows Phone.
- Emuladores Windows Phone 8.1: Paquete de emuladores de Windows Phone que agrega seis imágenes de emuladores a Visual Studio y así poder probar cómo funcionan las aplicaciones.
- SOAP UI: Herramienta open source utilizada el testeo de web services.

Además de esto también hará falta instalar el componente de Windows Hyper-V, es una herramienta de virtualización creada por Microsoft, es necesario ya que el emulador de Windows Phone es una máquina virtual que corre sobre este software.

3.2. Conceptos básicos para desarrollar en Windows Phone

3.2.1. Componentes de una aplicación

Antes de empezar el desarrollo de la aplicación en Windows Phone se requiere analizar la estructura que siguen estas aplicaciones y los componentes de diseño que son utilizados.

Los componentes más destacados son:

- Windows Runtime [023]: Es el conjunto de APIs que Windows Phone ofrece a las aplicaciones, el cual incluye una interfaz binaria de aplicación (ABI). Los componentes de WinRT actúan como librerías y están escritas en C# o C++ nativo. Estos objetos pueden ser llamados por identificador (namespaces desde la app) o por contratos (Operaciones de intercambio requeridas por un dato de un datatype concreto). Los objetos de WinRT son una capa encima del modelo del Component Object Model (COM). En resumen las API de WinRT permiten la interacción con los diferentes lenguajes de programación facilitando así el desarrollo.

- Layout Model [024]: Es utilizado en Windows Phone para diseñar la interfaz gráfica, controles y eventos. Estos elementos se definen bajo un lenguaje declarativo derivado del XML, llamado Extensible Application Markup Language (XAML). Además el SDK proporciona a los desarrolladores diferentes temas de diseño, incluyendo colores, controles, transiciones, animaciones e iconos.

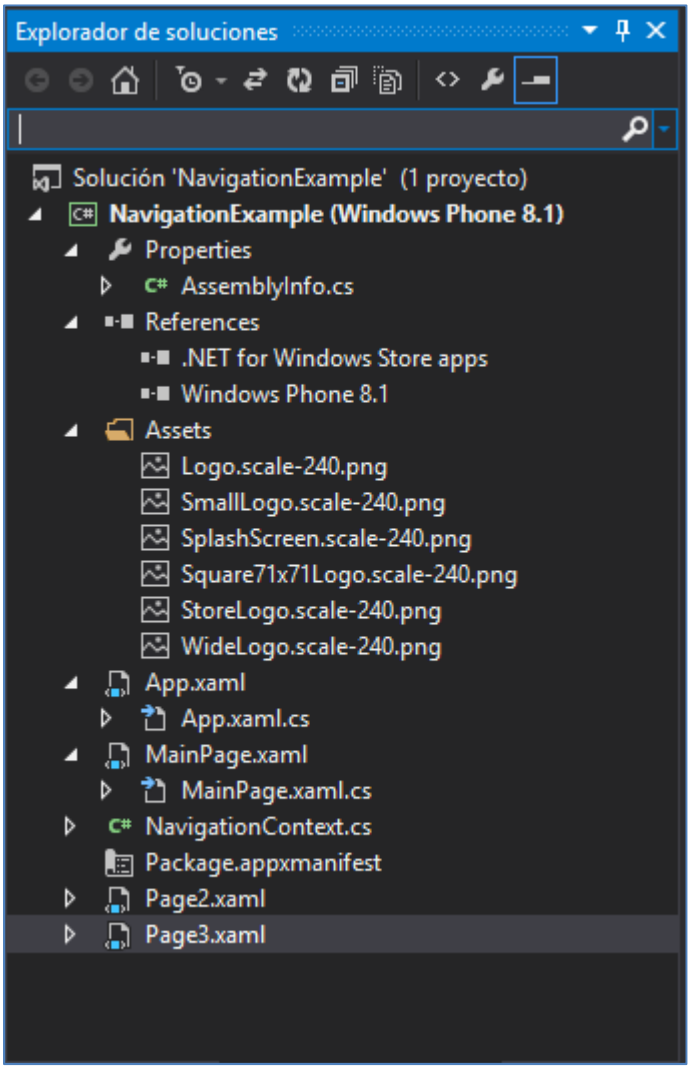
- App Manifest [025]: Cuando descargamos una aplicación, la tienda de Windows Phone proporciona un resumen de qué permisos requiere la aplicación. Esta información además de otras propiedades y configuraciones se almacenan en el fichero app manifest (package.appxmanifest.xml). Por ejemplo, opciones como el idioma por defecto, soporte para la instalación en la memoria externa, hardware adicional (NFC o cámara), etc. Cuando Visual Studio crea el package se incluye este fichero.

- App packaging: El resultado del proceso de construcción de una aplicación de Windows Phone es un paquete appx o appxbundle. Los paquetes appx son directorios zippeados que incluyen el manifest, ficheros, assets y contratos. Tras publicar el package en la tienda, Microsoft realiza algunas optimizaciones tal y como generación de código para mejorar el tiempo de arranca así como rasgos de seguridad contra la piratería.

3.2.2. Estructura de un proyecto Windows Phone

El desarrollo de aplicaciones para Windows Phone 8.1 llega de la mano de Visual Studio 2013.

Tras crear un nuevo proyecto, el IDE de Microsoft generará los archivos necesarios para que el proyecto funcione, estos ficheros están divididos en diferentes grupos y se pueden acceder a ellos desde el *Explorador de soluciones* tal y como se puede observar en la siguiente figura:



8: Detalle del Explorador de soluciones de Virtual Studio

De estas carpetas, es necesario tener claro el objetivo de las siguientes:

- Properties: El apartado de propiedades permite modificar los ajustes definidos en las plantillas (ficheros .cs) de nuestro proyecto activo.

- Referencias: Las referencias son utilizadas en Visual Studio para anexionar librerías adicionales al proyecto en el que estamos trabajando. Por ejemplo, una de las referencias añadidas al proyecto de la figura anterior, *.Net for Windows Store apps*, proporcionará acceso a un subconjunto nuevo de objetos relacionados con la interfaz de usuario, I/O, seguridad, networking, etc. Objetos específicos para apps que posteriormente van a ser publicadas en la Windows Store.

- Assets: Algunas aplicaciones van a necesitar de ficheros multimedia externos, estos ficheros se van agregar al proyecto dentro de la carpeta Assets, por defecto Windows Phone crea una serie de recursos visuales como iconos o pantalla de inicio.

- Sources: Aunque no están bajo una carpeta concreta, en la figura anterior tras el apartado Assets, he ido generando los ficheros fuente de la app. En una app cualquiera encontraremos como mínimo la pareja de ficheros MainPage.xaml y MainPage.xaml.cs. El diseño de la interfaz de usuario (diseño de los objetos, transiciones, eventos) de la página de inicio se gestionará desde la MainPage.xaml y la implementación código C# subyacente de ese diseño se gestionará en el fichero MainPage.xamls.cs. Este será un paradigma habitual en cada una de las páginas que forme la app.

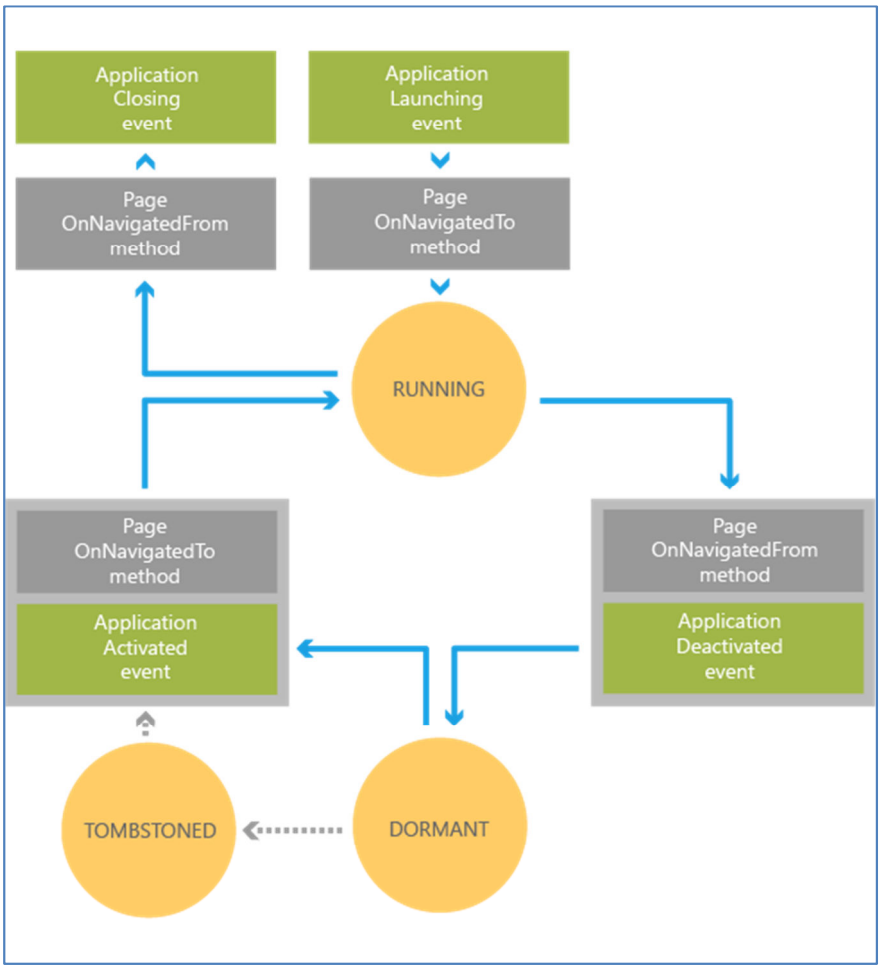
Además de los elementos anteriormente descritos, en todo proyecto realizado para Windows Phone existe un fichero básico, del que antes ya he hablado. Es el Package.appxmanifest, un documento XML en el que se van a definir una serie de características de la aplicación tales como su identificador, las dependencias del paquete, los recursos físicos del dispositivo que necesita y los elementos visuales que incorpora la app, por poner algunos ejemplos.

3.2.3. Ciclo de vida de una app

En este apartado pretendo describir el ciclo de vida de una aplicación de Windows Phone y cómo debería gestionarse su activación y desactivación [027].

En Windows Phone, solo una aplicación se ejecuta en primer plano en tiempo real. Esta obligación se toma para asegurar que la aplicación dispone de los recursos necesarios para ejecutarse de una manera fluida. Ya que una aplicación está ejecutándose en primer plano, cuando el usuario cambia hacia otra app, esta es suspendida o desactivada, dependiendo del contexto y de la manera en la que el usuario ha realizado la acción.
El modelo de ejecución de Windows Phone proporciona una serie de eventos y API's relacionadas que permiten a las apps gestionar la activación y desactivación de manera que proporcione una experiencia de usuario consistente e intuitiva.

La siguiente figura ilustra el ciclo de vida de una app [026]. En este diagrama, los círculos son los estados de la aplicación. Los rectángulos muestran los eventos de la aplicación o página donde se deben gestionar los estados.



9: Ciclo de vida de una app Windows Phone.

Guiándonos por la figura anterior, el ciclo de vida de cualquier aplicación se iniciará tras la selección de la misma desde la lista de apps instaladas, ya sea desde su propio Tile, desde notificaciones o cualquier otra forma. Cuando la app se ejecute, se mostrará la pantalla de arranque, cosa que indica que una nueva instancia ha sido ejecutada. Es posible mostrar información contextual que haga conocer al usuario detalles de interacciones anteriores con la app, tal como los últimos documentos vistos, pero no debería parecer como si el usuario volviera a una instancia previamente ejecutada de la app.
Así pues cuando una nueva instancia de la aplicación se ejecuta, se dispara el Application Launching Event. Para ayudar a que la app se cargue rápidamente, se debe tener en cuenta de no añadir excesivas tareas en el código del disparador del evento.

Una vez ejecutada, la aplicación está en modo Running y continuará estándolo hasta que el usuario se desplace adelante, salga de la app o se desplace a la página de inicio. No es necesario proporcionar un mecanismo para que el usuario salga o cierre la app.

El método OnNavigatedFrom es ejecutado una vez el usuario se desplaza de una página a otra de la app. Una vez el método es ejecutado, la aplicación almacena el estado de la

página para que pueda ser restaurada si el usuario vuelve a la página y ésta no está en memoria.

Otro evento se dispara cuando el usuario se desplaza adelante, fuera de la aplicación (al pulsar el botón de inicio) o ejecutando otra aplicación es la desactivación.
Al ejecutarse el evento, la app deberá guardar cualquier dato para que pueda ser recuperado posteriormente. Las apps de Windows Phone incorporan un objeto llamado State, es un diccionario que puede ser utilizado para almacenar el estado de la aplicación.
Es posible para la app terminar completamente tras la ejecución del evento. Cuando una aplicación se ha cerrado, no se conserva los datos del diccionario State. Así que se debe almacenar cualquier estado para que pueda ser utilizado como información entre instancias.

Al completarse el evento de desactivación el SO tratará de poner la app en un estado inactivo (Dormant). En este estado, todos los hilos de ejecución de la aplicación se detienen pero permanece intacta en memoria. Si la app se reactiva, no es necesario realizar nada para restablecerla, por qué su estado se conserva intacto.

Posteriormente, la app puede pasar a estar en estado Tombstoned (Suspendida) en este estado la app ha sido terminada, pero el SO aún conserva información sobre la navegación y diccionario de estado que se guardó durante el proceso de desactivación. Cada dispositivo mantendrá información relativa a la suspensión de hasta 5 apps. Si una está suspendida y el usuario quiere volver a ella, ésta será ejecutada de nuevo y la app puede utilizar la información de estado para volver al punto en el que se encontraba antes de ser suspendida.

Cuando un usuario recupera una aplicación en estado inactivo o suspendido, se ejecuta el evento de activación. La app debería verificar el valor de la propiedad IsApplicationInstancePreserved para determinar si se vuelve de un estado inactivo o suspendido y así actuar en consecuencia. Las apps no deberían ejecutar tareas intensas que consuman muchos recursos del dispositivo al ejecutarse este evento. Estas tareas pueden ejecutarse en segundo plano una vez ha cargado la aplicación o ya que los diccionarios de estado se guardaron durante el evento de Desactivación y están accesibles en memoria, se pueden utilizar para no saturar el sistema en operaciones de lectura y escritura.

El método OnNavigatedTo es llamado cuando el usuario se desplaza de una página a otra. Esto incluye cuando la app se ejecuta por primera vez, cuando el usuario navega entre páginas y cuando la app es relanzada tras haber estado en un estado inactivo o suspendido. En este método, la app puede revisar si la página es de una nueva instancia o no. Si no lo es, la información de estado no necesita ser recuperada, en caso contrario, se debería utilizar esta información para recuperar el estado de la interfaz de usuario.

Por último, el evento de finalización se genera cuando el usuario se desplaza hacia atrás desde la primera página de la app. En este caso, la app se cierra y no se guarda información de estado. No obstante el manejador del evento permite almacenar información que puede ser compartida por las instancias. El límite para almacenar la información de la app es de 10 segundos. Pasado ese periodo el SO finaliza la ejecución. Es por ello que es una buena idea guardar la información permanente durante la vida de la app y así evitar largas operaciones de escritura/lectura durante el evento de finalización.

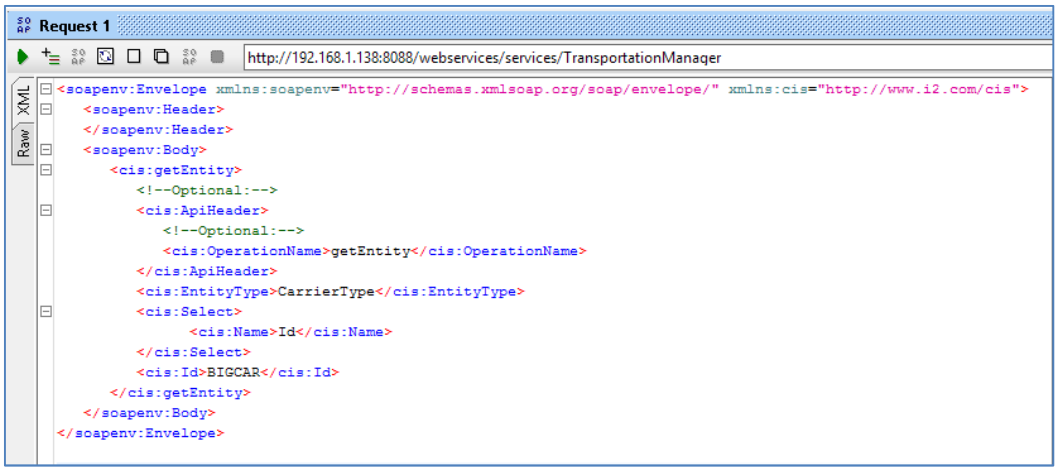
3.3. Conceptos básicos sobre SOAP-Web Services

Una de las partes principales de mi aplicación consiste en lograr implementar un cliente que envíe correctamente peticiones al Web Service proporcionado por JDA. Para ello es necesario entender en que consiste un SOAP Web Service.

SOAP es el acrónimo de Simle Object Access Protocol. Un protocolo de comunicaciones entre aplicaciones vía internet, aporta beneficios respecto otros protocolos de comunicación como RPC, ya que es independiente del SO, tecnología o del lenguaje de programación utilizado. Además al estar implementado sobre HTTP es aceptado por firewalls y proxis.

Un mensaje SOAP [017] básicamente es un documento XML que contiene los siguientes elementos:

- Un tag tipo Envelope que identifica el documento XML como un mensaje SOAP
- Un tag tipo Header que contiene la información de la cabecera
- Un tag tipo Body que contiene la información relativa a la petición o respuesta
- Un tag tipo Fault que contiene los errores o información de estado.



10: Ejemplo de mensaje SOAP

Conocemos los elementos básicos de cualquier mensaje SOAP, pero aún no sabemos cómo comunicarnos con el Web Service y es que es él quien va a publicar las funcionalidades y

cómo deben ser consumidas por el cliente. Toda esta información va a ser descrita en un documento con extensión .WSDL cuyo significado es Web Services Description Language. En este documento se especificará la ubicación así como las operaciones publicadas por el servicio.

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">
   <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" />
   <operation>
     <soap:operation soapAction="http://example.com/getTerm"/>
     <input><soap:body use="literal"/></input>
     <output><soap:body use="literal"/></output>
  </operation>
</binding>
```

11: Ejemplo de fichero WSDL

La figura anterior representa un ejemplo sencillo del contenido de un fichero .WSDL [019], en este ejemplo el elemento <PortType> podría ser comparado con una clase de un lenguaje de programación tradicional. La operación “getTerm” dispone de un mensaje de entrada llamado “getTermRequest” y uno de salida llamado “getTermResponse”.
Previamente estos mensajes han sido definidos con los argumentos que se necesitan y los datatypes correspondientes.

Ya que un servicio puede ser configurado para diferentes protocolos de comunicación, es necesario el elemento <Binding> para definir el protocolo y formato de datos utilizado.
En la anterior figura adecuamos al protocolo SOAP el PortType creado.

Todas estas normas están definidas por la W3C además de ser unas de sus recomendaciones para el desarrollo de aplicaciones Web desde el 2007.

3.4. Requisitos

3.4.1. Requisitos funcionales

- La aplicación permitirá consultar la información sobre el usuario conectado. Tal como su identificación, tipo de transportista, fecha de alta, fecha de expiración y dirección.
- La aplicación permitirá consultar la información relativa a los viajes (Identificador, servicio y estado) en estado aceptado por parte del usuario conectado.
- La aplicación permitirá consultar la información relativa a los viajes (Identificador, servicio y estado) en estado ofertado al usuario conectado en ese momento.
- La aplicación permitirá aceptar y rechazar un viaje ofertado al usuario conectado.
- La aplicación permitirá visualizar los detalles de un viaje tanto ofertado como aceptado por un usuario. Los detalles consisten en su ID, el número de pallets, distancia del viaje, volumen de la carga, número de paradas, origen y destino del viaje.
- La aplicación permitirá confirmar las paradas de un viaje con estado aceptado.
- La aplicación permitirá visualizar en un mapa el viaje, marcando en él las paradas del mismo.

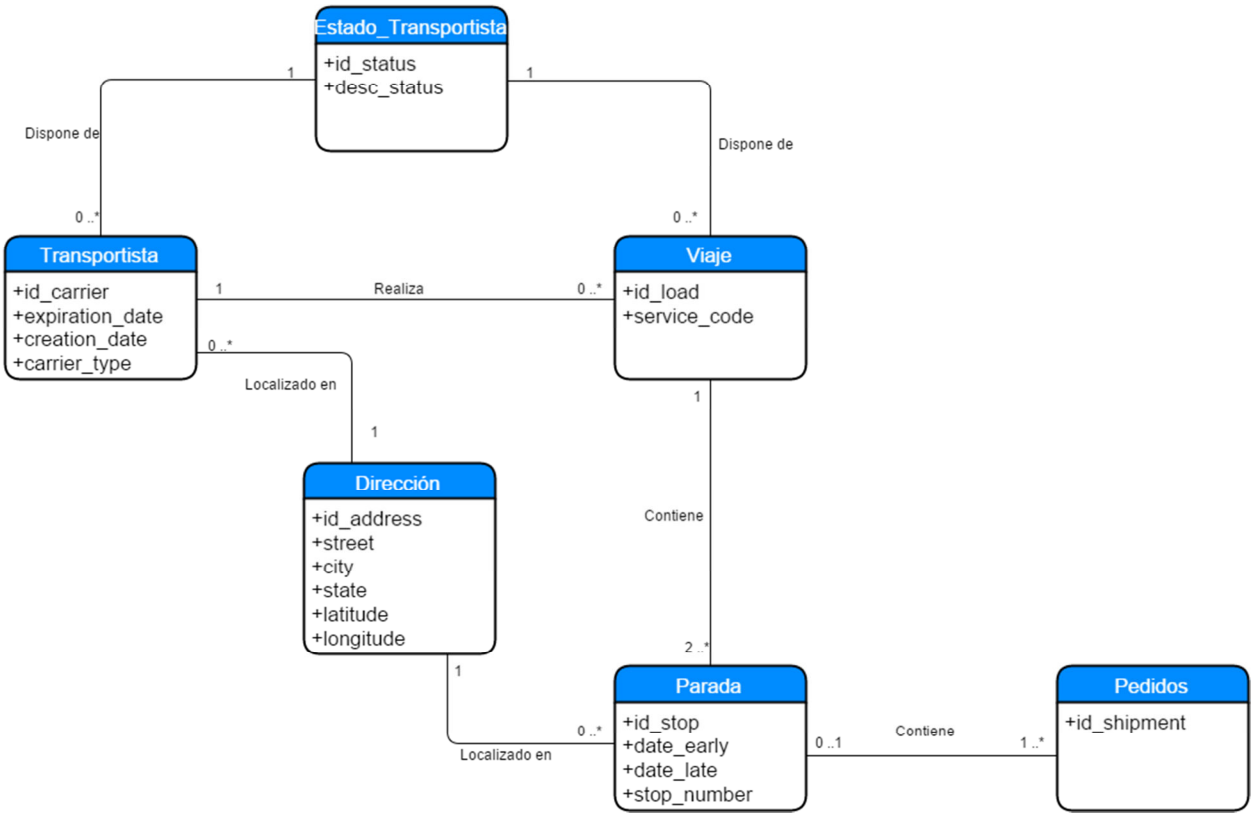
3.4.2. Requisitos no funcionales

- Dada la renovación de los dispositivos móviles del cliente, la aplicación deberá funcionar para el SO Windows Phone 8.1
- Como consecuencia del requisito anterior, el lenguaje para la construcción de la aplicación será XAML y C#.
- La tasa máxima de errores se ha estimado según la siguiente formula de errores/mdlc (miles de línea de código), en donde para la variable de errores se asignará el valor de 5, lo cual da como resultado 0,005 errores cada mil líneas.
- Se ha estipulado un tiempo máximo para las consultas de 3 segundos por consulta, y para las acciones que impliquen una modificación de los datos de 5 segundos.

3.5. Modelo conceptual

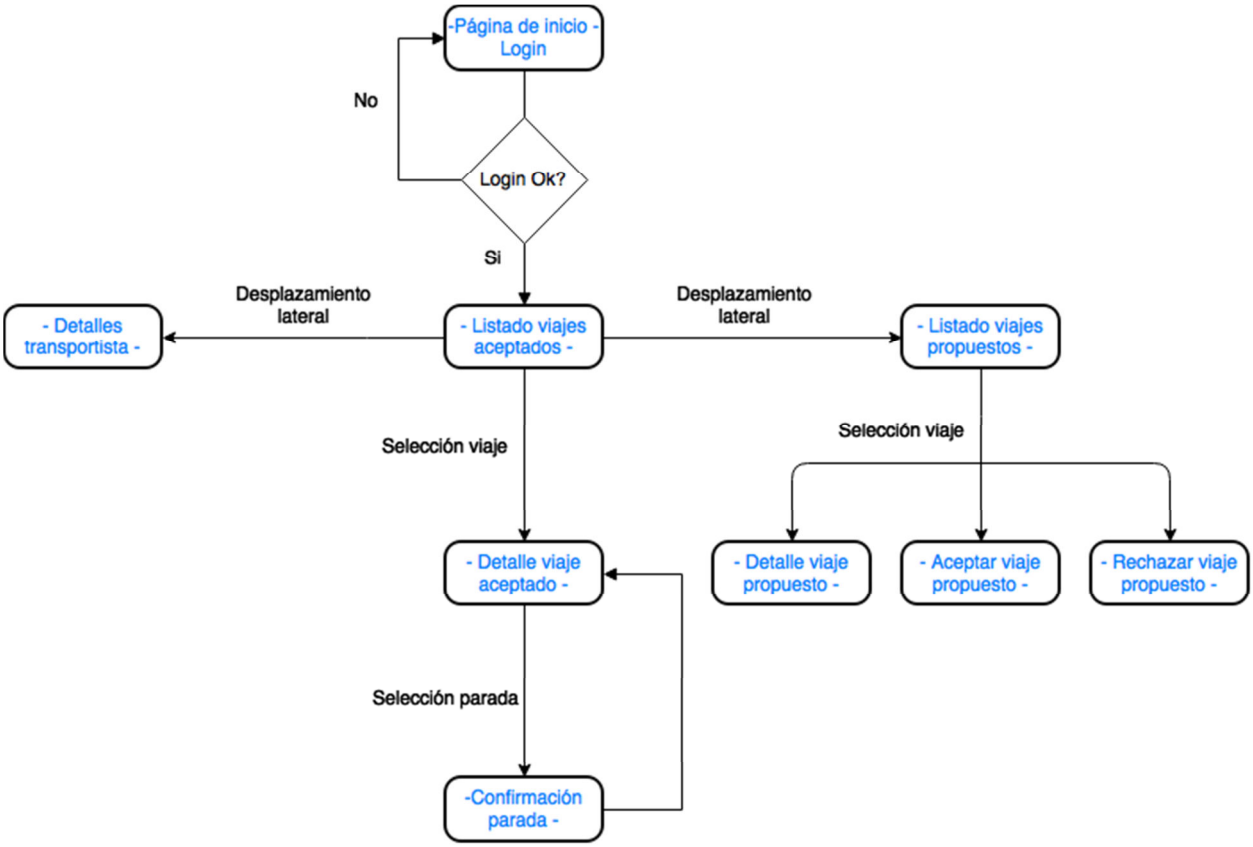
A partir del análisis de requisitos se ha podido diseñar el modelo conceptual, a través del cual, se muestran los conceptos que forman parte del dominio y las interrelaciones entre estos.
El MC mostrado es representado según el siguiente diagrama conceptual:

.



11: Diagrama del modelo conceptual

3.6. Flujo de información



12: Diagrama de flujo de la aplicación

3.7. Pantallas

En este apartado voy a tratar de describir el diseño de pantallas llevado a cabo para poder cubrir los requisitos funcionales anteriormente identificados.

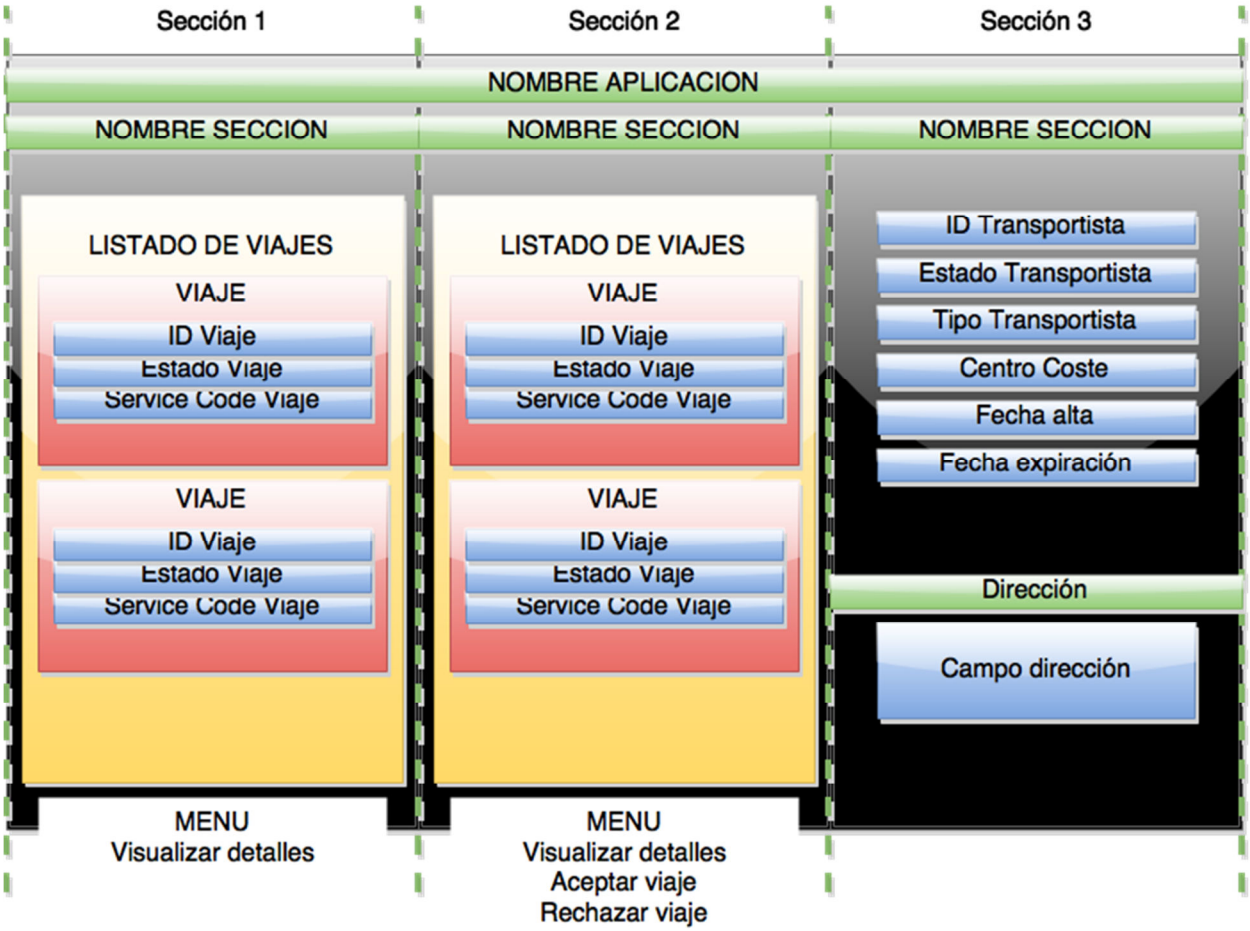
La aplicación está formada por 4 pantallas diferentes (aunque se considerarán 3 a efectos prácticos), la primera es la página de inicio donde se validará el ID del transportista. Una segunda página que se puede considerar como página principal, ya que es desde donde se accederá tanto el listado de viajes propuestos, como el listado de viajes aceptados y los detalles del transportista registrado.

Por último tendremos 2 pantallas muy parecidas, ya que ambas permitirán al usuario acceder a los detalles del viaje seleccionado aunque con ciertas acciones limitadas dependiendo del tipo de viaje a mostrar.

13: Diseño de la página inicial

En la figura anterior se muestra el diseño de la página de inicio, esta se compone de 2 bloques de texto en la parte superior, en ellos se mostrará el nombre de la aplicación y el nombre de la página respectivamente, bajo esta información se incluirá un campo de texto con un texto precargado que indica al usuario que escriba su ID de transportista, al marcar el campo automáticamente se borrará el texto y el usuario podrá escribir directamente el ID. Para hacer efectivo el proceso de validación estará disponible un botón encargado de inicializar esa petición.

Además de estos elementos, existirá un campo de texto oculto que solo se activa cuando hay un error en el proceso de validación, ya sea porque el ID es incorrecto o porque no puede conectar con el servidor.



14: Diseño de la página principal

Como se puede apreciar en la figura anterior, la visualización de la información referente a los viajes ofertados, aceptados y detalles de transportista se realizará desde una única pantalla, he elegido integrarlo así ya que el desarrollo de aplicaciones para Windows Phone libera algunos controles novedosos [029] como es el caso del control Hub [036], mediante este componente es posible estructurar la información en diferentes secciones, el acceso a las mismas es fácil e intuitivo haciendo scroll lateral y además se ha convertido en una seña de identidad de las aplicaciones de Windows Phone.

El diseño esta página estará formado por 3 secciones, una vez el usuario se ha validado correctamente en la aplicación, se procederá a cargar esta pantalla, de la cual inicialmente se muestra el contenido de la sección central.

Dicha sección está formada por un campo de texto donde se identifica el nombre de la aplicación (común para toda la pantalla) y otro campo de texto con el nombre de la sección cargada. Tras estos elementos, se creará un objeto tipo ListView [034], que es una lista deslizable horizontalmente de elementos, donde cada elemento de la lista será una colección de datos customizada, en este caso la colección de datos estará formada por 3 campos de texto: ID del viaje, Estado de viaje y Service Code del viaje.

Existirá un apartado de esta sección oculto inicialmente, se desbloqueará al seleccionar uno de los elementos de la lista, en ese momento aparecerá un menú emergente donde poder marcar las siguientes opciones: Aceptar viaje, Rechazar viaje o Visualizar detalles del viaje.

Al desplazarse hacia la izquierda, el usuario se puede visualizar la sección 1, el diseño de esta sección es prácticamente idéntico a la sección anterior ya que va a mostrar información referente a viajes de la misma manera, aunque la petición realizada y la selección de datos es diferente.
El elemento que difiere en ambas secciones es el menú emergente, ya que el estado en el que se encuentran los viajes hace que carezca de sentido aceptar o rechazar el viaje.

Por último analizaremos el diseño de la sección 3. Es posible acceder a ella realizando un desplazamiento lateral en ambas direcciones ya que el Hub por defecto se comporta como una lista circular.
El diseño de esta sección está basado en campos de texto en los que se muestra la información recogida del Web Service, e identificada por el cliente como requisitito tal y como el ID del transportista, su estado, que tipo de transportista es, la fecha en la que se dio de alta, en la fecha en la que expirará su contrato con la empresa.
Además de esta información, agregaré en un solo campo de texto la información relativa a la dirección del transportista (calle, número, ciudad, código postal).

Anteriormente hemos visto que una de las opciones disponibles en el menú es “Visualizar detalles”, haciendo clic en esta opción se mostrará una nueva pantalla. En la siguiente figura se puede ver el diseño aplicado.

15: Diseño de la página de detalles

En esta pantalla inicialmente, además del nombre de la aplicación y pantalla, se mostrará mediante diversos campos de texto, información detallada del viaje, como es el número de pallets, peso, distancia del viaje o volumen de la carga. Así como origen, destino del viaje, número de pedidos y número de paradas.

Tras estos elementos, he decidido crear mapa donde situar el origen, destino y en el caso que existan, paradas intermedias del viaje. Para ello será necesario incluir en el código el objeto MapControl, sobre el cual voy a poder indicar ubicaciones o trazar líneas entre distintos puntos.

En esta pantalla si el usuario se desplaza verticalmente, podrá visualizar también los detalles de las paradas para poder así confirmarlas si lo requiere.

Este apartado se ha implementado nuevamente mediante una lista de elementos, donde cada elemento de la ListView es un objeto customizado que se llamará parada. Este objeto está formado por los siguientes campos de texto como el identificador de la parada, el número de parada dentro del viaje, si existen pedidos para entregar o recibir, la fecha temprana a la que se espera la llegada del transportista, la fecha más tardía a la que puede llegar el transportista así como la dirección de la parada.

También se implementará un handler, para gestionar las acciones a realizar cuando el usuario selecciona alguno de los elementos de la lista, por ejemplo en ese momento aparece un menú emergente en el cual el usuario puede confirmar la parada, alterando además el estado de la parada en caso que no haya problema para completar la petición.

Comentar que el menú no estará disponible en caso que los detalles del viaje que se vaya a consultar sean de un viaje propuesto. Ya que no tiene transportista asignado.

Así pues este sería el diseño elegido para construir las 4 pantallas que forman la aplicación.

4. IMPLEMENTACIÓN

Este apartado tiene por objetivo exponer las partes más relevantes del código [030], sin entrar en excesivo detalle de otros apartados que, aunque son importantes para el buen funcionamiento de misma, son comunes en cualquier desarrollo, como pueden ser la inicialización, creación y asignación de memoria de objetos, gestión de botones, etc.

Para un pleno funcionamiento de esta aplicación es necesario como mínimo que el usuario de su visto bueno a una conexión a un servidor remoto, así como a acceso a la ubicación actual ya sea de un sensor GPS integrado en el dispositivo o de la información de red disponible:

```
<?xml version="1.0" encoding="utf-8"?>
<Package xmlns="http://schemas.microsoft.com/appx/2010/manifest"
xmlns:m2="http://schemas.microsoft.com/appx/2013/manifest"
xmlns:m3="http://schemas.microsoft.com/appx/2014/manifest"
xmlns:mp="http://schemas.microsoft.com/appx/2014/phone/manifest">
  <Identity Name="60bbc688-a84c-4d7e-961b-2fce312cfa34" Publisher="CN=David"
Version="1.0.0.0" />
  <mp:PhoneIdentity PhoneProductId="60bbc688-a84c-4d7e-961b-2fce312cfa34"
PhonePublisherId="00000000-0000-0000-0000-000000000000" />
  <Properties>
    <DisplayName>TMS_v1</DisplayName>
    <PublisherDisplayName>David</PublisherDisplayName>
    <Logo>Assets\StoreLogo.png</Logo>
  </Properties>
  <Prerequisites>
    <OSMinVersion>6.3.1</OSMinVersion>
    <OSMaxVersionTested>6.3.1</OSMaxVersionTested>
  </Prerequisites>
  <Resources>
    <Resource Language="x-generate" />
  </Resources>
  <Applications>
    <Application Id="App" Executable="$targetnametoken$.exe" EntryPoint="TMS_v1.App">
      <m3:VisualElements DisplayName="TMS_v1" Square150x150Logo="Assets\Logo.png"
Square44x44Logo="Assets\SmallLogo.png" Description="TMS_v1" ForegroundText="light"
BackgroundColor="transparent">
        <m3:DefaultTile Wide310x150Logo="Assets\WideLogo.png"
Square71x71Logo="Assets\Square71x71Logo.png">
        </m3:DefaultTile>
        <m3:SplashScreen Image="Assets\SplashScreen.png" />
      </m3:VisualElements>
    </Application>
  </Applications>
  <Capabilities>
    <Capability Name="internetClientServer" />
    <DeviceCapability Name="location" />
  </Capabilities>
</Package>
```

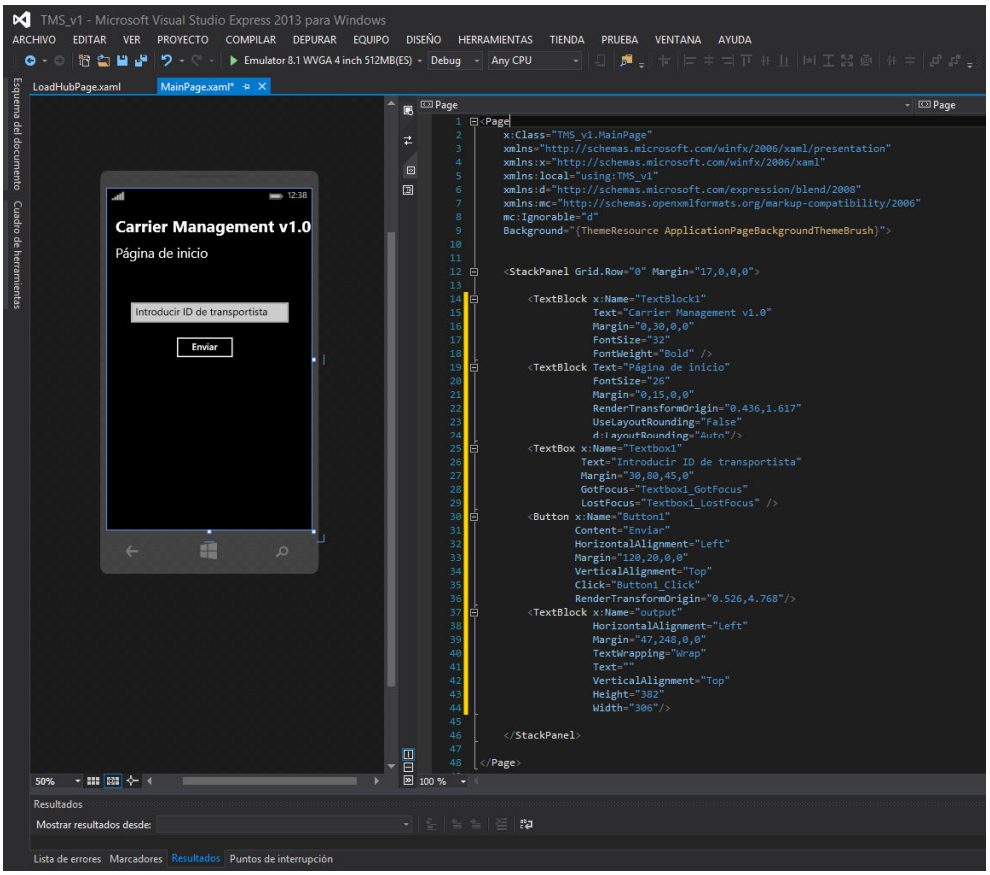
4.1. Estructura de la aplicación

Como he descrito anteriormente, la aplicación estará formada por 4 pantallas.
De lo que se desprende que el desarrollo estará formado por 4 elementos del tipo *Page*, cada *Page* es una clase de C# que es heredada del objeto *Frame*, este objeto es la parte visible para el usuario, y sobre el *Frame* se irá cargando cada una de los diferentes objetos *Page*, con tal de que estén disponibles como una nueva ventana de la aplicación.

Además de los elementos visibles, la aplicación tiene un claro objetivo que es el de actuar como un cliente SOAP [033], es por ello que habrán algunas clases no visibles al usuario pero que son básicas para el funcionamiento, como serán la clase Requests (Listado de peticiones a realizar al Web Service), la clase HttpClient (Envío de peticiones http) y la clase XDocument (Documento XML).

4.1.1. Pantalla inicial

Esta será la pantalla desde donde se validará que el usuario es correcto y a partir de aquí dará paso a la pantalla donde se gestionarán todas las acciones disponibles de los usuarios.
Tal y como he expuesto en anterioridad, en los desarrollos de Windows Phone se suele dividir por una parte la creación, colocación de los objetos dentro del layout haciendo uso del lenguaje XAML, ya que permite por un lado ver los resultados del diseño en tiempo real y además ofrece una sintaxis más fácil y rápida de interpretar.
En la figura siguiente se puede observar el código utilizado para el diseño de la página principal.



16: Detalle de Visual Studio, implementación del diseño

En concreto bajo el elemento *Page*, del cual hay una serie de propiedades ya predefinidas y que no se deben alterar, he añadido un objeto *StackPanel*, que es uno de los controles para el layout disponibles, dentro del cual he agregado los elementos de la página, en este caso 2 objetos *TextBlock*, los cuales he utilizado para mostrar el nombre de la aplicación y el de la página, además de 1 *TextBox* en el que se capturará el ID del usuario y 1 *Button*, cuya definición aparece en la siguiente figura:

```
30          <Button x:Name="Button1"
31                  Content="Enviar"
32                  HorizontalAlignment="Left"
33                  Margin="120,20,0,0"
34                  VerticalAlignment="Top"
35                  Click="Button1_Click"
36                  RenderTransformOrigin="0.526,4.768"/>
37          <TextBlock x:Name="output"
38                     HorizontalAlignment="Left"
39                     Margin="47,248,0,0"
40                     TextWrapping="Wrap"
41                     Text=""
42                     VerticalAlignment="Top"
43                     Height="382"
44                     Width="306"/>
```

17: Código XAML del botón y TextBox

Como se puede observar, aparte de definir propiedades como el identificador de cada elemento(*x:Name*), texto o márgenes, se han definido directamente desde la vista diseño diferentes eventos. En el momento que se define un evento, directamente se genera la cabecera del método en el fichero .cs donde se implementará toda la lógica de cada pantalla.

El siguiente ejemplo del evento *Button1_Click* servirá para analizar el código derivado y otros componentes básicos de la aplicación:

```
private async void Button1_Click(object sender, RoutedEventArgs e)
{
    //Creamos el objeto Uri
    Uri resourceAddress;

    // El valor de 'AddressField' es asignado como variable global. Si no se puede generar una URI válida,
    // se notificará el error y no se continua con el programa.
    if (!Helpers.TryGetUri(AddressField, out resourceAddress))
    {
        output.Text = "Servidor no disponible en estos momentos";
        return;
    }
    else
    {
        try
        {
            Requests Request1 = new Requests(carrier);
            HttpClient httpClient = new HttpClient();
            //Añadimos caberecera SOAPAction, si no el servidor no reconoce el mensaje como SOAP
            httpClient.DefaultRequestHeaders.Add("SOAPAction", "http://xml.foo.com/MyService/v1/Query");
            HttpResponseMessage response = await httpClient.PostAsync(resourceAddress, new HttpStringContent(Request1.CarrierValidate()));

            String xml_content = response.Content.ToString();
            //Xdocument montamos el mensaje de respuesta como XML
            XDocument doc = XDocument.Parse(xml_content);
            XNamespace ns = "http://www.i2.com/cis";
            foreach (XElement element in doc.Descendants(ns + "CompletedSuccessfully"))
            {
                if (element.Value == "true")
                {
                    Frame.Navigate(typeof(LoadHubPage), carrier);
                }
                else
                {
                    output.Text = "Transportista no válido";
                }
            }
        }
        catch (Exception oEx)
        {
            // Manejamos las excepciones... No hay mucho que hacer en este punto
            output.Text = "Servidor no disponible. Contacte con el deptartamento técnico.\n\n Codigo error: " + oEx.Message;
        }
    }
```

18: Código del evento Button_Click del fichero MainPage.xaml.cs

Repasando el código del evento, en primer término se inicializa un objeto de la clase *URI*, el objetivo es determinar si la URL a la que apunta al Web Service está bien construida. En caso negativo, se notificaría al usuario y se procedería a salir del programa.
Una vez la dirección es correcta, se creará un nuevo objeto de la clase *Requests*, en esta clase se agrupan los diferentes tipos de mensajes que pueden ser enviados al Web Service, además se inicializará un objeto tipo *HttpClient* el cual tiene por objetivo transmitir dicho mensaje al servidor.

La clase *Requests* toma como argumento el ID que el usuario ha introducido previamente y en base a este valor se generará la siguiente petición:

```
public string CarrierValidate()
{
    return "<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\" xmlns:cis=\"http://www.i2.com/cis/\">"
      + "<soapenv:Header>"
      + "</soapenv:Header>"
      + "<soapenv:Body>"
      + "<cis:getEntity>"
      + "<!--Optional:-->"
      + "<cis:ApiHeader>"
      + "<!--Optional:-->"
      + "<cis:OperationName>getEntity</cis:OperationName>"
      + "</cis:ApiHeader>"
      + "<cis:EntityType>CarrierType</cis:EntityType>"
      + "<cis:Id>" + CarrierID + "</cis:Id>"
      + "</cis:getEntity>"
      + "</soapenv:Body>"
      + "</soapenv:Envelope>"
      + "";
}
```

19: Detalle del método CarrierValidate, mensaje SOAP

Así pues este mensaje, tras indicar en la cabecera que será un tipo de mensaje SOAP (*httpClient.DefaultRequestHeaders.Add("SOAPAction","http://xml.foo.com/MyService/v1/Query");*) Estará listo para ser enviado al servidor mediante el método *PostAsync*, como su nombre indica es un método asíncrono, este tipo de métodos requieren que el bloque desde el que se lanza el método este marcado con el modificador async (Ver el figura 17).

Este es a priori el punto donde es más probable que se pueda generar una excepción, ya sea porque el mensaje no está bien construido o por algún problema de conectividad. El caso es que al generarse una excepción esta será capturada en el apartado *catch (Exception oEx)* y se mostrará por pantalla tanto el mensaje de error como el código de error.

Otro de los puntos básicos para la creación de la aplicación que aparecen en esta pantalla, es el de saber interpretar correctamente el mensaje de respuesta del Web Service. La respuesta, inicialmente un mensaje HTTP, se transformará a un objeto de la clase *XDocument*, es la manera que he encontrado más adecuada ya que ofrece al desarrollador diferentes instrucciones de procesamiento que me permitirán manejar de manera más simple los nodos del documento XML.

Así pues, una vez construido el objeto *XDocument*, el código busca la etiqueta *<CompletedSuccessfully>* en el mensaje de respuesta ya que el valor de dicho etiqueta (true o false) indicará si el transportista se encuentra dado de alta o no en el TMS.

En el caso satisfactorio, se procederá a navegar a la siguiente página, en Windows Phone esto se realiza mediante la instrucción *Frame.Navigate(typeof(LoadHubPage), carrier).* Donde el primer argumento es la página a que queremos dirigirnos y como segundo argumento he decidido enviar el ID del transportista, ya que será necesario en futuras peticiones.

4.1.2. Pantalla Hub

De la segunda pantalla de la aplicación voy a describir 2 aspectos que he considerado destacables, por una parte la interfaz de usuario implementa el control Hub, un control especifico de Windows Phone y por otra parte, como se trabaja con listas de elementos.

El control Hub permite estructurar en diferentes secciones una pantalla de la aplicación, haciendo sencilla la navegación entre las diferentes secciones mediante un desplazamiento lateral. En la siguiente figura se puede observar el código XAML de una sección de la pantalla.

```
<Grid x:Name="LayoutRoot">
    <Hub x:Name="Hub1" x:Uid="Hub" DefaultSectionIndex="1">
        <Hub.HeaderTemplate>
            <DataTemplate>
                <TextBlock Text="Carrier Management v1.0" Margin="0,30,0,0" FontSize="32" FontWeight="Bold" />
            </DataTemplate>
        </Hub.HeaderTemplate>
        <HubSection x:Uid="HubSection1"
                    x:Name="HubSection1"
                    Header="Viajes aceptados"
                    Width="400"
                    d:LayoutRounding="Auto">
            <DataTemplate x:Name="DataTemplate1">
                <ListView x:Name="LoadAcceptedGrid"
                          Grid.Row="1"
                          ItemsSource="{Binding}"
                          Margin="0,0,17,0"
                          Tapped="LoadAcceptedGrid_Tapped">
                    <ListView.ItemContainerStyle>
                        <Style TargetType="ListViewItem">
                            <Setter Property="HorizontalContentAlignment" Value="Stretch"/>
                            <Setter Property="BorderBrush" Value="LightGray" />
                            <Setter Property="BorderThickness" Value="0,3,0,3" />
                        </Style>
                    </ListView.ItemContainerStyle>
                    <ListView.ItemTemplate>
                        <DataTemplate>
                            <StackPanel>
                                <TextBlock Text="{Binding ID}" Margin="0,15,0,0" FontWeight="Bold" FontSize="24" TextWrapping="Wrap" VerticalAlignment="Top"/>
                                <TextBlock Text="{Binding Status}" FontSize="18" TextWrapping="Wrap" VerticalAlignment="Top"/>
                                <TextBlock Text="{Binding ServiceCode}" FontSize="18" TextWrapping="Wrap" VerticalAlignment="Top"/>
                            </StackPanel>
                        </DataTemplate>
                    </ListView.ItemTemplate>
                </ListView>
            </DataTemplate>
        </HubSection>
```

20: Código XAML de la seccion1 del Hub

Para esta pantalla primero he definido un control *Grid*, es una alternativa más flexible que otros controles para el layout, ya que permite la colocación de elementos en diversas filas o columnas, de manera que para situar listas customizadas es más adecuada que, por ejemplo, *StackPanel*.

Dentro de este *Grid*, he creado el controlador para el *Hub*. No se han definido apenas modificadores, tan solo que la sección en la que se va a posicionar por defecto sea la 1, ya que he pensado que sería la sección que más van a utilizar los usuarios.
Tras la definición del *Hub*, he modificado la cabecera (*Hub.HeaderTemplate*) para que muestre el nombre de la aplicación, dándole la ubicación y formato que he creído más adecuado.

Llegado a este punto ya es posible ir introduciendo las diferentes secciones que conforman el *Hub*. Concretamente en la figura anterior se muestra la sección *HubSection1*, como en otros elementos, se definen inicialmente algunos modificadores para que el comportamiento sea el esperado y tras esto, bajo etiqueta *DataTemplate*, cuyo objetivo es el de permitir mostrar una colección de datos sin un formato estándar, he añadido el control *ListView*.
La idea general es que a la hora de implementar una lista de elementos, primero se defina la estructura de los elementos en el fichero XAML y además se realice un enlace de datos (Binding) para poder vincular los valores obtenidos del Web Service a cada elemento de la *ListView*.

En la siguiente figura es posible observar el código utilizado para implementar esta idea.

```
private void PopulateAcceptedLoads(string c, XDocument doc)
{

    XNamespace ns = "http://www.i2.com/cis";

    foreach (var element in doc.Descendants(ns + "findEntitiesResponse"))
    {

        foreach (XElement childEllement in element.Descendants(ns + "Load"))
        {
            Accepteditems.Add(new LoadClass()
            {
                ID = "Id: " + childEllement.Element(ns + "Id").Value.ToString(),
                Status = "Estado: " + ReplaceLoadStatus(childEllement.Element(ns + "CurrentLoadOperationalStatusEnumVal").Value.ToString()),
                ServiceCode = "Servicio: " + childEllement.Element(ns + "ServiceCode").Value.ToString()

            });
        }
    }
}
```

21: Detalle del método PopulateAcceptedLoads

Como se puede observar, he creado un método llamado *PopulateAcceptedLoads*, este recibe por argumento la respuesta del Web Service ya en formato XML. En este caso el código busca dentro del XML los viajes con estado aceptado para un transportista en concreto, esto se obtiene extrayendo cada elemento identificado con la etiqueta <*Load*>.

Para cada una de estos viajes, se declara e inicializa un nuevo objeto de la clase *LoadClass* y se agrega el objeto a una lista según la siguiente sentencia:

List<LoadClass> Acceptedítems = new List<LoadClass>()

```
public class LoadClass
{
    public string ID { get; set; }
    public string ServiceCode { get; set; }
    public string Status { get; set; }
}
```

22: Clase LoadClass

Una vez se han agregado todos los elementos que cumplen con los criterios de búsqueda del XML a la lista *Acceptedite*ms, esta está preparada para ser vinculada a la *ListView* creada en el diseño de la página. Para conseguir tal propósito, he hecho uso de la propiedad *DataContext* que permite establecer el contexto de datos de un elemento del Framework, en este caso concreto bastaría con la siguiente instrucción:

```
HubSection1.DataContext = Acceptedeitems;
```

Estos serían los aspectos que he encontrado más significativos a la hora de implementar un control *Hub* y una *ListView*.

4.1.3. Pantalla Detalles

La pantalla desde la que se visualizan los detalles del viaje introduce un elemento que he considerado interesante exponer, este elemento es el *MapControl* [037] que va a aportar múltiples mejoras a la aplicación, ya sea pudiendo geoposicionar la ubicación del transportista, marcando en el mapa la ubicación de las paradas del viaje o trazando líneas entre estas paradas.

En la siguiente figura se expone el código desarrollado para incluir en el diseño de la página el objeto *MapControl*.

```
<Grid>
        <Maps:MapControl
    x:Name="MapWithRoute"
    MapServiceToken="AouQGxro1eQUirnHwoTLAH81K3jPN8pql7W8qGdmbPhIsge7M_PCBMGlZZ9ZKQH5"
    Height="320"
    Width="350"
    Margin="-15,20,0,20" />
</Grid>
```

23: Diseño MapControl

Previamente, para poder hacer uso del mapa hay que añadir en la cabecera del fichero XML la siguiente declaración:

```
using Windows.UI.Xaml.Controls.Maps;
```

Una vez cumplidos esos dos puntos, el mapa ya se podría visualizar desde la aplicación, aunque además de visualizar el mapa, he considerado de utilidad para los transportistas el situar en el mapa las paradas de un viaje en concreto y trazar las líneas entre los puntos.

```
void addNewPolyline()
{
    List<BasicGeoposition> PosList = new List<BasicGeoposition>();
    List<MapIcon> MapIconList = new List<MapIcon>();
    MapPolyline line = new MapPolyline();
    foreach (var data in StopPosition)
    {
        PosList.Add(new BasicGeoposition()
        {
            Latitude = double.Parse(data.latitude),
            Longitude = Convert.ToDouble(data.longitude)
        });
        MapIconList.Add(new MapIcon(){
            Location = new Geopoint(new BasicGeoposition()
        {
            Latitude = Convert.ToDouble(data.latitude),
            Longitude = Convert.ToDouble(data.longitude)
        }),
        Title = "Parada",
        NormalizedAnchorPoint = new Point(0.5, 1.0)
        });

    }

    foreach (var Icon in MapIconList)
    {
        MapWithRoute.MapElements.Add(Icon);
    }

    line.Path = new Geopath(PosList);
    line.StrokeThickness = 8;
    MapWithRoute.Center = new Geopoint(PosList[0]);
    MapWithRoute.ZoomLevel = 9;
    MapWithRoute.MapElements.Add(line);
}
```

24: Detalle del método addnewPolyline

Para conseguir este objetivo, he creado un nuevo método llamado *addNewPolyline*, el cual en primer término va a recorrer la lista de paradas obtenida del Web Service al inicializar la pantalla.
De cada parada se dispone de la latitud y longitud almacenada en el TMS, valores que servirán para poder crear tanto un nuevo objeto de la clase *BasicGeoposition*, como otro de la clase *MapIcon* [038].

Tras registrar estos elementos, los agregamos al Mapa mediante la instrucción *MapWithRoute.MapElements.Add(Icon);*

Finalmente se asigna al objeto de la clase *MapPolyLine*, que servirá para poder dibujar la línea entre los puntos del mapa, la ubicación de los puntos y por último se definen diversas propiedades relacionadas con el objeto *MapControl*, tales como centrar el mapa, establecer el zoom aplicado a la zona o el estilo de la línea.

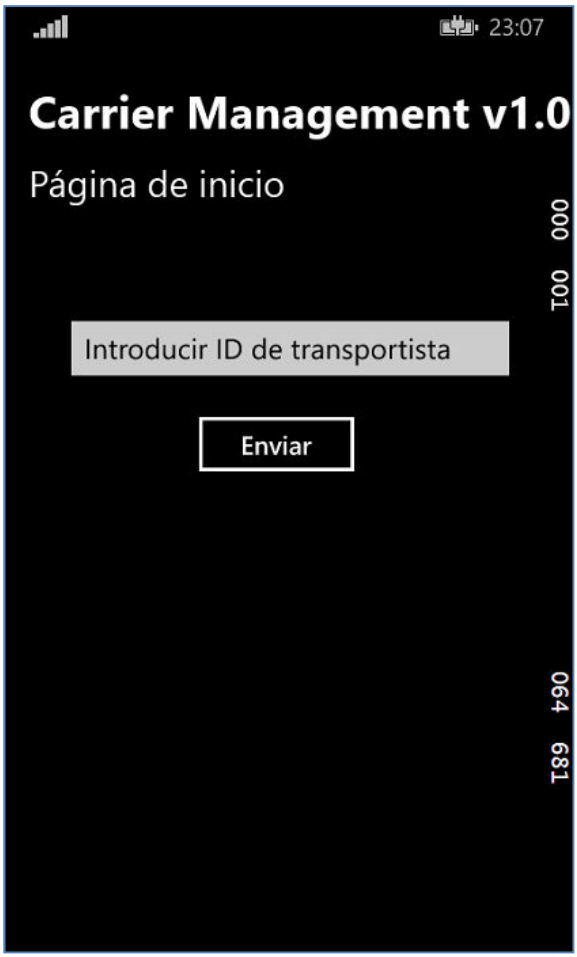
4.1.4. Resultados y pruebas

Según los objetivos mínimos que me había marcado, el resultado ha sido el esperado. El usuario va a poder navegar por la aplicación de una manera correcta, fluida y sencilla.

Durante la fase de construcción, he ido verificando que el rendimiento de la aplicación sea aceptable, principalmente en lo referente a la comunicación con el Web Service, ya que podría ser un cuello de botella en un entorno real.

Además, en lo referente a la navegación, el usuario puede consultar sin problemas aparentes tanto sus datos, como datos sobre los viajes aceptados, propuestos, detalles de las paradas de los viajes así como ubicar los viajes en el mapa. A continuación se muestran algunos ejemplos de cada una de las funcionalidades de la aplicación:

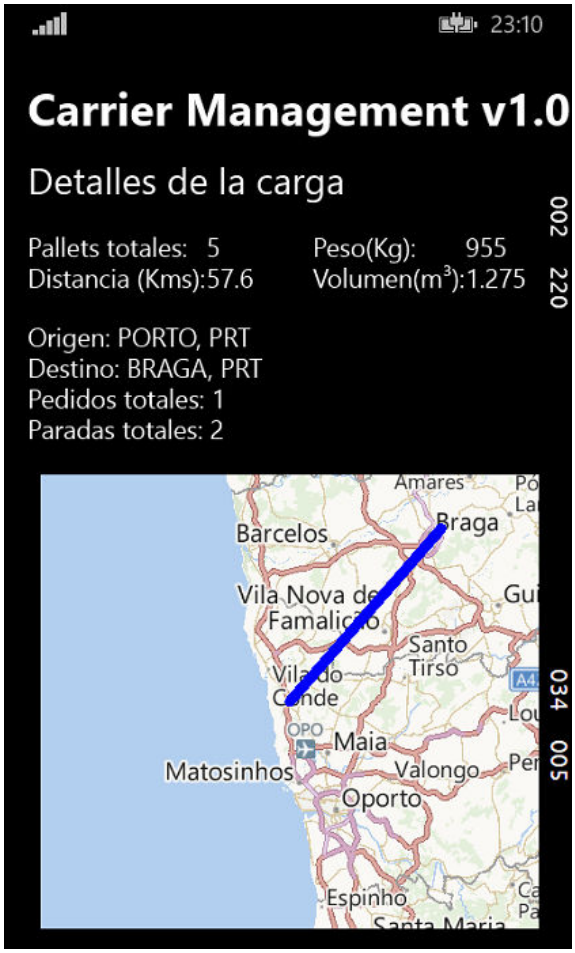
Visualizar detalles viaje aceptado:



25: Pantalla de inicio

26: Listado viajes aceptados

27: Selección viaje 23113



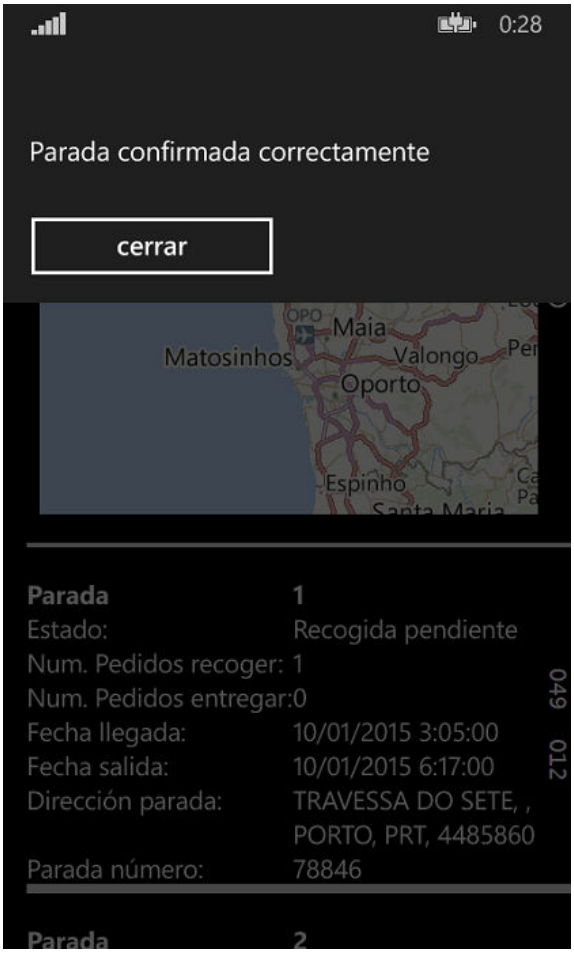
28: Detalles de la carga (Viaje)

Confirmar Parada:

29: Selección viaje 23113

30: Visualización detalles (Paradas)

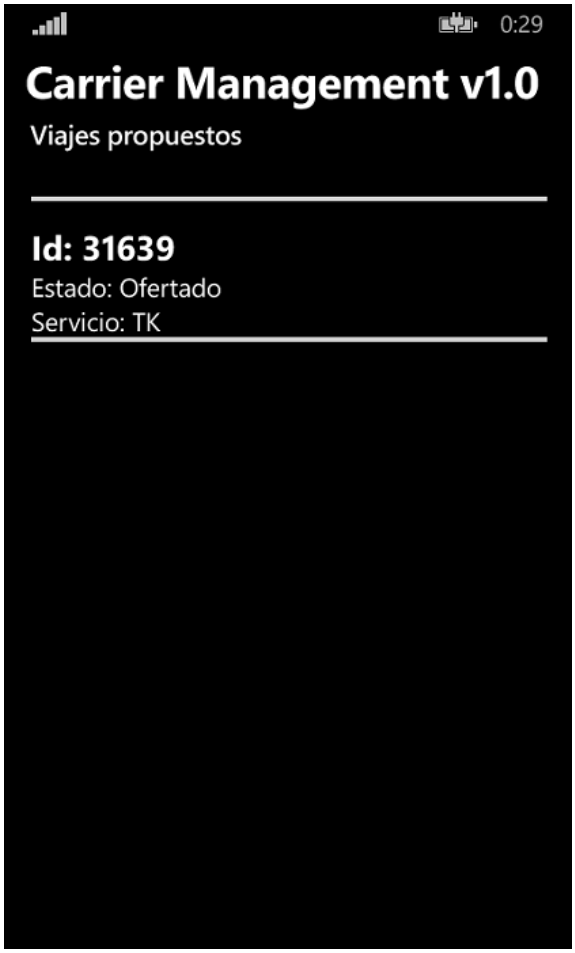
31: Selección parada 1



32: Ventana emergente confirmando parada

33: Vuelta a la pantalla detalles, el estado de la parada ha cambiado a recogida confirmada

Rechazar viaje:

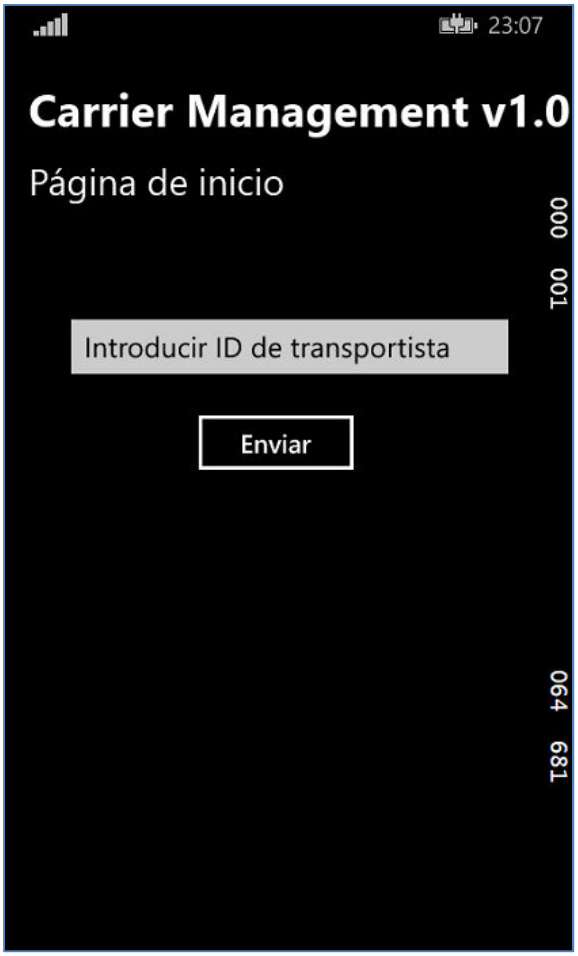


34: Sección viajes propuestos

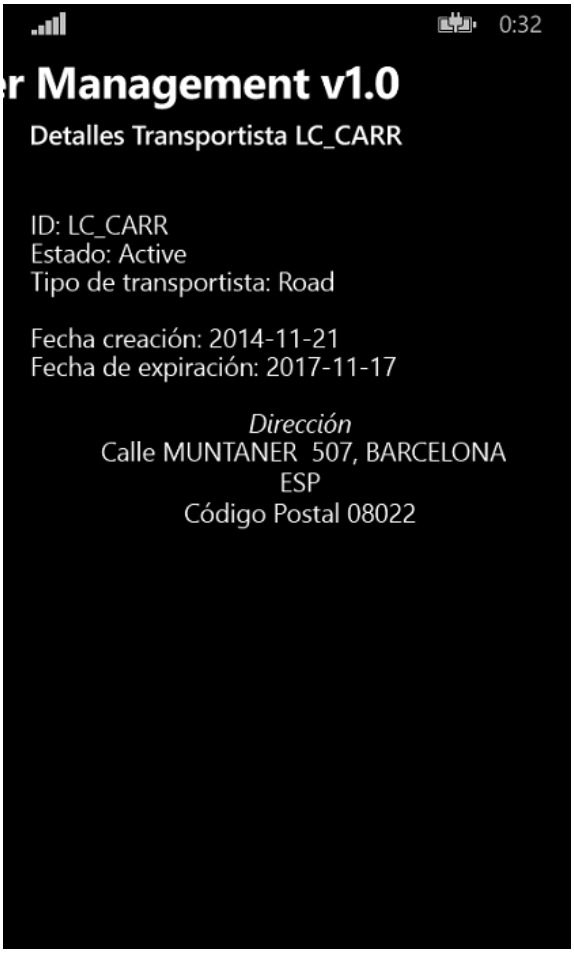
35: Selección viaje 31639

36: Mensaje emergente, carga (Viaje) rechazado

Visualizar detalles transportista:



37: Pantalla de inicio (Login)



38: Sección con los detalles sobre el transportista

5. PLANIFICACIÓN Y ANÁLISIS ECONÓMICO

Para la realización de este proyecto, se ha dividido el trabajo en las siguientes etapas:

- Análisis de requisitos: Definir los objetivos del proyecto, conseguir obtener la visión completa por parte del cliente/usuario de que funcionalidades debe cubrir la aplicación así como eliminar posibles ambigüedades en los requisitos proporcionados.

- Formación: Aunque en un proyecto de ingeniería del software no estaría incluida, al ser un proyecto académico he necesitado de una etapa de autoformación e investigación para recopilar información tanto de las herramientas de programación, como conocer las tecnologías que se han aplicado en el desarrollo y conceptos relacionados con los TMS.

- Diseño y arquitectura: Determinar cómo funcionará de forma general sin entrar en detalles propios de la implementación, incorporando consideraciones sobre las herramientas encontradas, decidiendo finalmente cuales utilizar. Incluye el diseño de los componentes del sistema.

- Implementación: Se traduce el diseño a código. Es la etapa donde he empezado a obtener resultados “tangibles”.

- Pruebas: Analizar que la aplicación realiza correctamente las tareas que he indicado en el diseño. Además corregir los errores que van apareciendo.

- Documentación: Elaboración de la memoria.

5.1. Costes

Llegar a cuantificar los costes que supone el llevar a cabo este proyecto es una tarea de cierta complejidad, es por ello que se va a dividir los costes relativos a licencias, de los costes relacionados con las horas efectivas de trabajo.

5.1.1. Licencias

Antes de pasar a determinar el número y en algunos casos el precio de las licencias necesarias, he considerado conveniente explicar las necesidades que han llevado a la creación de la aplicación.
Este proyecto académico nace para cubrir la supuesta necesidad que pueden tener ciertas empresas que han decidido implantar la solución TMS de JDA o que ya lo han hecho , y

que quieren poner a disposición de sus transportistas una aplicación móvil que pueda darles acceso en cualquier parte a información de relevancia. Con esto pretendo poner sobre la mesa que no se concibe la creación de este proyecto sin disponer del software de JDA.

El modo de licenciamiento de JDA tiene en cuenta diversos aspectos como el tipo de servidor en el que se va a instalar, el número de usuarios planificadores o el número de cargas a planificar por día. En coste promedio de una licencia para la suite de JDA Transportation Manager es de 12.000 euros, además se firma un contrato de mantenimiento anual.
A su vez, el TMS de JDA va a requerir de una base de datos Oracle. El coste de la licencia de Oracle fundamentalmente va a depender del tipo de edición que se vaya a instalar, JDA Transportation Manager está certificado para la Standard o Enterprise edition.
Además del tipo de edición que se decida instalar, el coste vendrá determinado según una de las dos métricas de licenciamiento elegida por el cliente, o bien por procesadores o bien por usuarios nominativos. Como es un asunto complejo no voy a determinar un coste promedio de esta licencia.

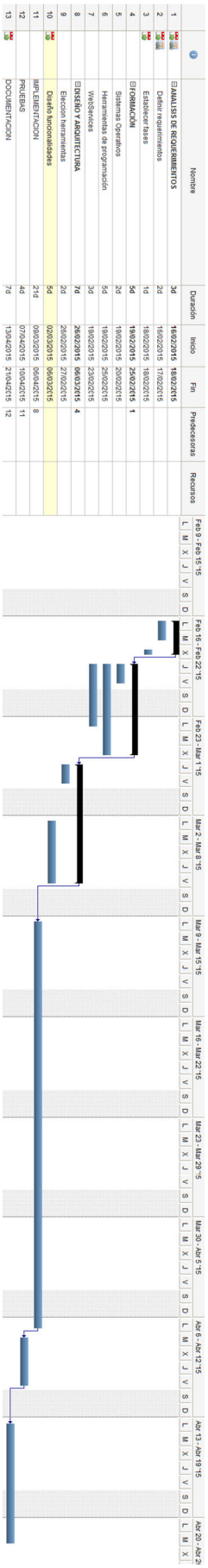
Por último necesitaremos una licencia de desarrollador para Windows Phone. El coste de la licencia es de 19 euros para cuentas individuales y 99 para cuentas de empresa.

5.1.2. Horas de trabajo

Partiendo de la base que todo el trabajo lo realizaría el mismo recurso, por ejemplo un analista/programador junior y que el precio hora fuera de 21 euros. El coste para cada una de las etapas del proyecto sería:

Análisis de requisitos	3 días	504 €
Definir requisitos	2 días	336 €
Establecer fases	1 día	168 €
Diseño y arquitectura	7 días	1176 €
Elección herramientas	2 días	336 €
Diseño funcionalidades	5 días	840 €
Implementación	20 días	3360 €
Pruebas	4 días	672 €
Documentación	7 días	1176 €
TOTAL PROYECTO	51 días	6888 €

5.2. Diagrama de Gantt



6. CONCLUSIONES Y POSIBLES MEJORAS

Una vez completado todo el trabajo, mi valoración sobre el proyecto es que he cumplido favorablemente con los objetivos marcados.
Hay que tener en cuenta que he retomado el proyecto final de carrera tras unos años apartado del mundo académico y en el que mi aventura profesional no se está enfocando en áreas relacionadas con el desarrollo de software, es por ello que este proyecto me ha servido para poner en práctica muchos de los conocimientos adquiridos durante la carrera.

También quería dejar constancia que detrás de esta aplicación (más o menos compleja) hay muchos otros conocimientos sobre áreas relacionadas con la informática. Para llegar a completar este proyecto he tenido por ejemplo, que crear y administrar una máquina virtual para Windows Server 2012, instalar un servidor de bases de datos Oracle, realizar tareas propias de un DBA para disponer de una base de datos, instalar y poner en marcha un servidor de aplicaciones Oracle Weblogic, etc.

Creo por ello que ha sido un proyecto en el que he tenido que aprender de múltiples áreas relacionadas con la informática.

Lógicamente que hayan tantos componentes involucrados ha hecho que haya invertido mucho tiempo en encontrar la fuente de información correcta, tarea que en diversas ocasiones no ha resultado fácil ya sea porque la calidad de la fuente no fue adecuada o incluso por una sobreinformación que ha costado filtrar convenientemente.

Me gustaría terminar las conclusiones poniendo un ejemplo de algo que siempre ha sido una constante en las carreras tecnológicas.
Cuando inicié el proyecto creí acertado que el desarrollo fuera para Windows Phone, por varios motivos ya comentados en la memoria. En ese momento la última versión era Windows Phone 8.1, anunciado en abril de 2014. Pues bien, poco más de un año después, ya se ha anunciado Windows 10 Mobile, que antes de finalizar el año se desplegará en todos los dispositivos que soportaban la anterior versión.
Con esto pretendo exponer el continuo esfuerzo de los desarrolladores por autoformarse, revisar las novedades en el SDK y adaptar sus aplicaciones ya no solo a nuevas plataformas si no a la revisión casi obligada ante cada nueva actualización de SO.

6.1. Problemas

Aunque en un principio la decisión de trabajar con C# y programar para Windows Phone suponía algo positivo ya que me puede ayudar profesionalmente, el desconocimiento tanto del IDE como de las clases y métodos específicos de la plataforma ha hecho que en ciertos

momentos el avance de la implementación se haya paralizado, o incluso en algunos casos, haya tenido que reescribir el código al darme cuenta que la idea inicial no se podía implementar en C#.

Además ha sido un poco confuso que hayan tantos cambios entre la API de Windows Phone 8.1 y la de Windows Phone 8. Muchas veces al buscar dudas sobre el determinado uso de una feature, por ejemplo cómo hacer uso de los mapas, la forma de realizarlo entre ambas versiones era muy distinta.

6.2. Limitaciones

A lo largo del desarrollo de la aplicación he ido anotando las limitaciones del mismo, las voy a exponer en el siguiente listado:

- La aplicación requiere de una conexión a internet para poder comunicarse con el Web Service.

- La aplicación no obtiene rutas entre dos puntos, hacer uso de Bing Maps podría mejorar esto.

- Dependiendo de los elementos que haya alrededor de un punto del mapa, no se visualiza el icono que representa una parada ya que Microsoft considera más importante la información almacenada sobre un punto que el icono que le dé el desarrollador.

- No se ha implementado una autentificación basada en usuario y password, con más tiempo de investigación se habría incluido.

6.3. Mejoras

Considerando tanto las limitaciones expuestas en el punto anterior, los conocimientos adquiridos durante el proyecto y conversaciones en el ámbito laboral, han surgido ideas que pueden mejorar la aplicación, estas son algunas:

- No se ha implementado el cálculo de rutas entre puntos ya que no hay una manera gratuita de conseguirlo, es necesario registrar la aplicación previo pago de la licencia de desarrollador. Sería una mejora para los transportistas disponer la ruta a seguir.

- Utilizar la geoposicion de los dispositivos para almacenarlas en el TMS, a la hora de montar los viajes, el sistema podría tomar esta información para asignar al transportista más cercano a la ubicación de origen del viaje para ofertárselo.

- Implementar la autentificación basada en usuario y password.

- Posibilidad de confirmar parcialmente una parada (Por ejemplo, si uno de los pedidos no estaba disponible a la hora de realizar la carga).

7. BIBLIOGRAFIA

[001] Sistemas Operativos [En línea] Los Angeles, California: Wikimedia Foundation. [Consulta: 21 de febrero de 2015] Disponible en: https://es.wikipedia.org/wiki/Sistema_operativo

[002] Mobile Operating System [En línea] Los Angeles, California: Wikimedia Foundation. [Consulta: 21 de febrero de 2015] Disponible en: https://en.wikipedia.org/wiki/Mobile_operating_system

[003] iOS [En línea] Los Angeles, California: Wikimedia Foundation. [Consulta: 21 de febrero de 2015] Disponible en: https://en.wikipedia.org/wiki/IOS

[004] Android [En línea] Los Angeles, California: Wikimedia Foundation. [Consulta: 22 de febrero de 2015] Disponible en: https://es.wikipedia.org/wiki/Android

[005] Windows Phone [En línea] Los Angeles, California: Wikimedia Foundation. [Consulta: 22 de febrero de 2015] Disponible en: https://es.wikipedia.org/wiki/Windows_Phone

[006] Wordwide Smartphone Sales to End Users Q42014 [En línea] Egham, UK: Gather. [Consulta: 22 de febrero de 2015] Disponible en: http://www.gartner.com/newsroom/id/2996817

[007] Smartphone OS Market Share [En línea] Framingham, USA: IDC Corporate. [Consulta: 24 de febrero de 2015. Disponible en: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[008] Android Studio [En línea] California, USA: Google Inc. [Consulta: 24 de febrero de 2015]. Disponible en: https://developer.android.com/sdk/index.html

[009] XCode IDE [En línea] California, USA: Apple Inc. [Consulta: 24 de febrero de 2015]. Disponible en: https://developer.apple.com/xcode/

[010] Transportation Management System [En línea] Los Angeles, California: Wikimedia Foundation. [Consulta: 1 de marzo de 2015] Disponible en: https://en.wikipedia.org/wiki/Transportation_management_system

[011] Magic quadrant for transportation management system [En línea] Connecticut, Estados Unidos: Gartner Inc. [Consulta: 2 de marzo de 2015] Disponible en: http://blog.ikiseleva.com/wp-content/uploads/2014/07/magic_quadrant_for_transport_2014.pdf

[012] SaaS [En línea] Los Angeles, California: Wikimedia Foundation. [Consulta: 2 de marzo de 2015] Disponible en: https://es.wikipedia.org/wiki/Software_como_servicio

[013] Oracle Transportation Management Knowledge Zone [En línea] California, USA: Oracle Corporation. [Consulta: 2 de marzo de 2015] Disponible en: http://www.oracle.com/partners/esa/products/applications/transportation-mgmt/get-started/index.html

[014] Oracle Transportation Management Data Sheet [En línea] California, USA: Oracle Corportaion. [Consulta: 2 de marzo de 2015] Disponible en: http://www.oracle.com/us/products/applications/046953.pdf

[015] SAP Transportation Management Highlights [En línea] Weinheim, Alemania: SAP AG. [Consulta: 4 de marzo de 2015] Disponible en: http://www.sap.com/spain/solution/lob/scm/software/transportation-management/index.html

[016] Transportation Manager Brochure [En línea] Arizona, USA: JDA Software Group. [Consulta: 4 de marzo de 2015] Disponible en: http://www.jda.com/view/scm-brochure/JDA-Transportation-Manager/

[017] Web Services Example [En línea] Noruega: Refsnes Data. [Consulta: 5 de marzo de 2015] Disponible en: http://www.w3schools.com/Web Services/ws_example.asp

[018] Web Services Tutorial [En línea] Noruega: Fefsnes Data. [Consulta: 5 de marzo de 2015] Disponible en: http://www.w3schools.com/Web Services/

[019] WSDL [En línea] Los Angeles, California: Wikimedia Foundation. [Consulta: 2 de marzo de 2015] Disponible en: https://es.wikipedia.org/wiki/WSDL

[020] SOAP Introduction [En línea] Noruega: Fefsnes Data [Consulta: 5 de marzo de 2015] Disponible en: http://www.w3schools.com/Web Services/ws_soap_intro.asp

[021] Fusion Middleware Installation Guide for Oracle WebLogic Server [En línea] California, USA: Oracle Corporation. [Consulta: 20 de marzo de 2015] Disponible en: http://docs.oracle.com/cd/E12839_01/doc.1111/e14142/toc.htm

[022] Install Hyper-V [En Línea] Columbia, USA: Microsoft Corporation. [Consulta: 20 de marzo de 2015] Disponible en: https://technet.microsoft.com/en-us/library/hh846766.aspx

[023] Windows Phone Runtime [Blog] StackOverflow [Consulta: 20 de marzo de 2015] Disponible en: http://stackoverflow.com/questions/15463161/what-is-the-windows-phone-runtime

[024] Layout for Windows Phone 8 [En línea] Columbia, USA: Microsoft Corporation. [Consulta: 20 de marzo de 2015] Disponible en: https://msdn.microsoft.com/en-us/library/windows/apps/jj207042(v=vs.105).aspx

[025] Application Manifests [En línea] Columbia, USA: Microsoft Corporation. [Consulta: 20 de marzo de 2015] Disponible en: https://msdn.microsoft.com/en-us/library/aa374191(v=vs.85).aspx

[026] Ciclo de vida de la aplicación [En línea] Columbia, USA: Microsoft Corporation. [Consulta: 20 de marzo de 2015] Disponible en: https://msdn.microsoft.com/es-es/library/windows/apps/Hh464925.aspx

[027] App activation and deactivation for Windows Phone 8 [En línea] Columbia, USA: Microsoft Corporation. [Consulta 21 de marzo de 2015] Disponible en: https://msdn.microsoft.com/en-us/library/windows/apps/ff817008(v=vs.105).aspx

[028] Windows Phone 8.1 –platform concepts and app development [En línea] Dresden, Alemania: Sebastian Mueller. [Consulta 13 de marzo de 2015] Disponible en: http://www.academia.edu/7712500/Windows_Phone_8.1_platform_concepts_and_app_development

[029] Windows Phone 8.1 Development for absolute beginners [En línea] Redmond, Columbia: Bob Tailor. [Consulta: 7 de mayo de 2015] Disponible en: https://channel9.msdn.com/Series/Windows-Phone-8-1-Development-for-Absolute-Beginners

[030] C# Fundamentals Development for Absolute Beginners [En línea] RedMond, Columbia: Bob Tailor. [Consulta: 7 de mayo de 2015] Disponible en https://channel9.msdn.com/Series/C-Sharp-Fundamentals-Development-for-Absolute-Beginners/Series-Introduction-01

[031] Application Program Interface Guide for Transportation Manager [En línea] Arizona, USA. JDA Software Group and Inc. [Consulta: 1 de abril de 2015]

[032] Requerimientos funcionales y no funcionales: Requerimientos funcionales y no funcionales. [Blog] Ingenieriadesoftware [Consulta: 5 de mayo de 2015] Disponible en: http://ingenieriadesoftware.bligoo.com.mx/requerimientos-funcionales-y-no-funcionales-rf-rnf

[033] Call SOAP Web Service [Blog] StackOverflow [Consulta: 7 de mayo de 2015] Disponible en: http://stackoverflow.com/questions/28048506/call-soap-ws-from-windows-phone-8-1

[034] ListView, DataBinding and ItemTemplate [Blog] WPF-Tutorial [Consulta: 9 de mayo de 2015] Disponible en: http://www.wpf-tutorial.com/listview-control/listview-data-binding-item-template/

[035]Inicio rápido: Agregar controles ListView y GridView (XAML) [En línea] Columbia, USA: Microsoft Corporation. [Consulta: 9 de mayo de 2015] Disponible en: https://msdn.microsoft.com/es-es/library/windows/apps/xaml/Hh780650.aspx

[036] Guidelines for the hub control [En línea] Columbia, USA: Microsoft Corporation. [Consulta: 11 de mayo de 2015] Disponible en: https://msdn.microsoft.com/en-us/library/windows/apps/dn449149.aspx

[037] How to display maps in the MapControl [En línea] Columbia, USA: Microsoft Corporation. [Consulta: 11 de mayo de 2015] Disponible en: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/dn642089.aspx

[038] Windows Phone 8.1 for developers – Geolocation and Geofencing [Blog] JayWay: Andreas Ekberg [Consulta: 12 de mayo] Disponible en: http://www.jayway.com/2014/04/22/windows-phone-8-1-for-developers-geolocation-and-geofencing/